

centos7.2 ELK5.6 入门

| | |
|--|----|
| 一、环境设置..... | 3 |
| 2.centos7 把网卡名设置为 ethx..... | 3 |
| #1.修改网卡参数..... | 3 |
| #2.禁用该可预测命名规则..... | 4 |
| #3.重新生成 GRUB 配置并更新内核参数 | 4 |
| 3.其它配置..... | 4 |
| #1.关闭 selinux、防火墙（包括开机自启动） | 4 |
| #2.时间同步..... | 4 |
| #3.设置主机名..... | 4 |
| #4.epel 源安装..... | 5 |
| #5.修改文件连接数..... | 5 |
| 一、ELK 安装及简单配置 | 5 |
| 1.ELK 介绍..... | 5 |
| #1.核心组成..... | 5 |
| #2.四大组件..... | 6 |
| #3.ELK 工作流程..... | 6 |
| 2.安装 jdk1.8..... | 7 |
| 3.安装 ELK5.6..... | 8 |
| #1.ELK 安装（yum） | 8 |
| #2.设置环境..... | 9 |
| 4.elasticsearch 简单配置及启动..... | 9 |
| #1.查看安装相关..... | 9 |
| #2.修改配置..... | 10 |
| #3.启动..... | 10 |
| 5.logstash 简单配置及启动 | 11 |
| #1.查看安装相关..... | 11 |
| #2.修改配置..... | 11 |
| #3.配置 pipeline 文件（只是一个测试配置，获取 messages 信息） | 12 |
| #4.测试..... | 12 |
| #5.启动..... | 13 |
| 6.kibana 简单配置及启动..... | 13 |
| #1.查看安装相关..... | 13 |
| #2.修改配置..... | 13 |
| 7. x-pack 插件 | 14 |
| #1.给 elk 都安装上 x-pack 插件..... | 14 |
| #2.修改 logstash 配置 | 14 |
| #3.为 logstash 配置添加用户名和密码(为测试添加密码) | 15 |
| #4.配置 x-pack | 16 |
| #5.登陆认识一下界面..... | 16 |
| #登陆界面..... | 17 |
| #修改密码..... | 17 |
| #开发工具 Dev Tools（对 ES 中 API 操作都在这里） | 18 |

| | |
|---|----|
| 二、测试..... | 18 |
| 1.导入官方数据..... | 18 |
| 2.在 kibana 添加设置数据索引 | 19 |
| #1.添加有日期索引 | 19 |
| #2.创建普通索引 | 20 |
| 三、Elasticsearch RESTful API 操作（文档） | 21 |
| 0.相关知识点..... | 21 |
| #1.Elasticsearch 中的_source、_all、store 和 index 属性..... | 21 |
| 1.用 put 创建索引 | 22 |
| 例子 1..... | 22 |
| 例子 2..... | 24 |
| #创建索引的其它选项..... | 25 |
| #1）防止重复 ID（op_type=create） | 25 |
| #2）设置超时时间(timeout) | 26 |
| 2.get 获取文档信息 | 26 |
| #1.简单获取..... | 26 |
| #2.获取多个文档（_mget） | 26 |
| #3.获取某 type 下所有 id(_search)..... | 27 |
| 3.Update 更新信息..... | 27 |
| #1.信息更新（_update） | 27 |
| 4. bulk 批量操作（不能美化操作） | 27 |
| 5.Delete 删除操作 | 28 |
| #1.DELETE 简单删除..... | 28 |
| #2. _delete_by_query 查询匹配删除 | 29 |
| 四、Elasticsearch RESTful API 操作（search） | 29 |
| 1._search 查找..... | 29 |
| #1.搜索所有信息..... | 29 |
| #2.在特定类型中进行搜索..... | 29 |
| #3.可用参数..... | 30 |
| 2.通过 URI Search 查找 | 31 |
| #1.测过 URI search 查找 | 31 |
| #2.常用参数..... | 31 |
| 3.请求主体查找..... | 32 |
| 五、elasticsearch 查询 DSL..... | 32 |
| 1.Match ALL 查询..... | 32 |
| 2.全文搜索..... | 32 |
| #1.匹配查询..... | 33 |
| #2.短语查询..... | 33 |
| #3.短语前缀匹配查询..... | 34 |
| #4.查询语句查询..... | 34 |
| 3.其它..... | 34 |
| 六、logstash 配置讲解 | 34 |
| 1.简单的例子..... | 34 |
| #1.创建一个简单的例子..... | 34 |

| | |
|-------------------------------|----|
| #2.配置文件的重要组成部分..... | 36 |
| 2.logstash 配置语法 | 36 |
| #1.语法..... | 36 |
| #2.区段..... | 36 |
| #3.数据类型..... | 36 |
| #4.字段引用(field reference)..... | 36 |
| #5.条件判断(condition)..... | 37 |
| #6.命令行参数..... | 37 |
| #7.设置文件..... | 38 |
| 3.input 配置 | 38 |
| #1.file 文件例子 | 39 |
| #2.Sdtin 例子 | 39 |
| #3.rsyslog..... | 40 |
| 4.codec 配置 | 41 |
| #1.json 数据例子 | 42 |
| #2. Multiline 合并多行数据例子 | 44 |
| 5.filter 配置 | 45 |
| #1.date | 46 |
| #2.grok 正则捕获 | 47 |
| #3.dissect | 48 |
| #4.其它..... | 49 |
| 6.output | 49 |
| #1.elasticsearch..... | 49 |
| #2.标准输出(Stdout) | 50 |
| 七、logstash 场景示例 | 51 |
| 八、kibana..... | 51 |
| 附录一、参考文章..... | 51 |

一、环境设置

1.实验环境及要求

| 主机名 | 外网 IP（NAT） | 内网 IP | 备注 |
|---|---------------|--------------|---|
| Vm1 | 192.168.56.11 | 192.168.3.11 | 1.系统 centos7.2 64 位 2.内存要求最低 2G，我这里用 2.5G 3.swap 分区空间大小不能为 0，否则 ES 启动不起来！ |
| 其它： 1. ES 和 logstash：要求 JDK 或 openJDK 版本不低于 1.8 2. ELK 版本为当前稳定版本 5.5，为了方便以后升级和维护采用 yum 安装 | | | |

2.centos7 把网卡名设置为 ethx

#1.修改网卡参数

#说白了就是把 enoxxx 形式改为 ethx 形式

#每个主机的网卡名不同根据情况修改一下

```
cd /etc/sysconfig/network-scripts/  
cp ifcfg-eno16777736 ifcfg-eno16777736.orig  
cp ifcfg-eno33554960 ifcfg-eno33554960.orig  
sed -i '/eno16777736/s#eno16777736#eth0#g' ifcfg-eno16777736  
sed -n '/eth/p' ifcfg-eno16777736  
mv ifcfg-eno16777736 ifcfg-eth0  
  
sed -i '/eno33554960/s/eno33554960/eth1/g' ifcfg-eno33554960  
sed -n '/eth/p' ifcfg-eno33554960  
mv ifcfg-eno33554960 ifcfg-eth1
```

#2.禁用该可预测命名规则

#你可以在启动时传递 “net.ifnames=0 biosdevname=0 ” 的内核参数

```
grep 'GRUB_CMDLINE_LINUX' /etc/default/grub  
sed -i '/GRUB_CMDLINE_LINUX/s/rhgb/rhgb net.ifnames=0 biosdevname=0/' /etc/default/grub  
grep 'GRUB_CMDLINE_LINUX' /etc/default/grub
```

#3.重新生成 GRUB 配置并更新内核参数

#运行命令 grub2-mkconfig -o /boot/grub2/grub.cfg 来重新生成 GRUB 配置并更新内核参数

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

#重启

```
shutdown -r now
```

3.其它配置

#1.关闭 selinux、防火墙（包括开机自启动）

```
systemctl stop firewalld  
systemctl disable firewalld  
sed -i '/^SELINUX=/s/enforcing/disabled/' /etc/selinux/config  
grep '^SELINUX=' /etc/selinux/config  
setenforce off
```

#2.时间同步

```
yum install ntp -y  
/usr/sbin/ntpdate pool.ntp.org  
echo '#time sync by hua'>>/var/spool/cron/root  
echo '*/* * * * * /usr/sbin/ntpdate pool.ntp.org >/dev/null 2>&1'>>/var/spool/cron/root  
crontab -l
```

#3.设置主机名

```
cat /etc/hostname
```

#如果想起个名字可以用下面命令永久生效

```
hostnamectl set-hostname vm1
```

#4.epel 源安装

```
rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

#5.修改文件连接数

Linux 服务器查看服务器默认的 tcp 连接数命令是 `ulimit -n`，阿里云默认的 tcp 连接数是 65535，超过会有影响，我这里也采用阿里云的方式，如果物理机其实可以设置大 10 倍也没问题。

```
cat>>/etc/security/limits.conf<<EOF
#by hua
* soft nofile 65535
* hard nofile 65535
EOF
cat /etc/security/limits.conf
ulimit -n
```

一、ELK 安装及简单配置

1.ELK 介绍

ELK：开源分布式日志分析搜索平台

E：elasticsearch，它负责数据的存储和查询

L：logstash，它负责日志数据的过滤和解析

logstash 是 ELK 重要组成部分，日志进一步的处理差不多都需要经过 logstash 来完成的，除非自己开发 beats 将数据处理好直接写入到 elasticsearch 中。

K：Kibana，它负责 web 方式的前端展现

kibana 本质上是 elasticsearch web 客户端，是一个分析和可视化 elasticsearch 平台，可通过 kibana 搜索、查看和与存储在 elasticsearch 的索引进行交互。可以很方便的执行先进的数据分析和可视化多种格式的数据，如图表、表格、地图等。

这就形成了人们所说的：采集分析→数据存储→展现数据

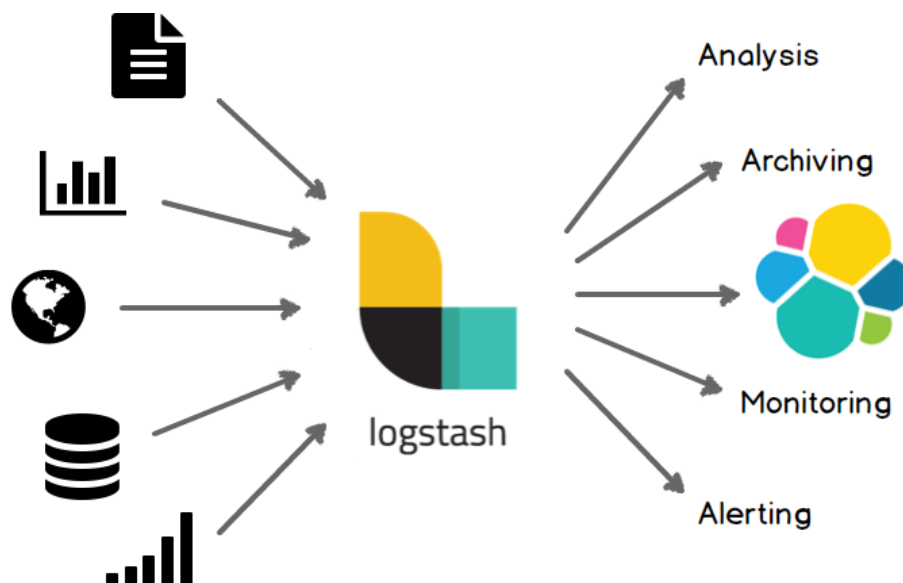
#1.核心组成

Elasticsearch：

一个开源分布式搜索引擎，它的特点有：分布式，零配置，自动发现，索引自动分片，索引副本机制，restful 风格接口，多数据源，自动搜索负载等。

Logstash：

一个完全开源的工具，它可以对你的日志进行收集、分析，并将其存储供以后使用



kibana :

一个开源和免费的工具，它可以为 Logstash 和 Elasticsearch 提供的日志分析友好的 Web 界面，可以帮助您汇总、分析和搜索重要数据日志。

#2.四大组件

Logstash: logstash server 端用来搜集日志;

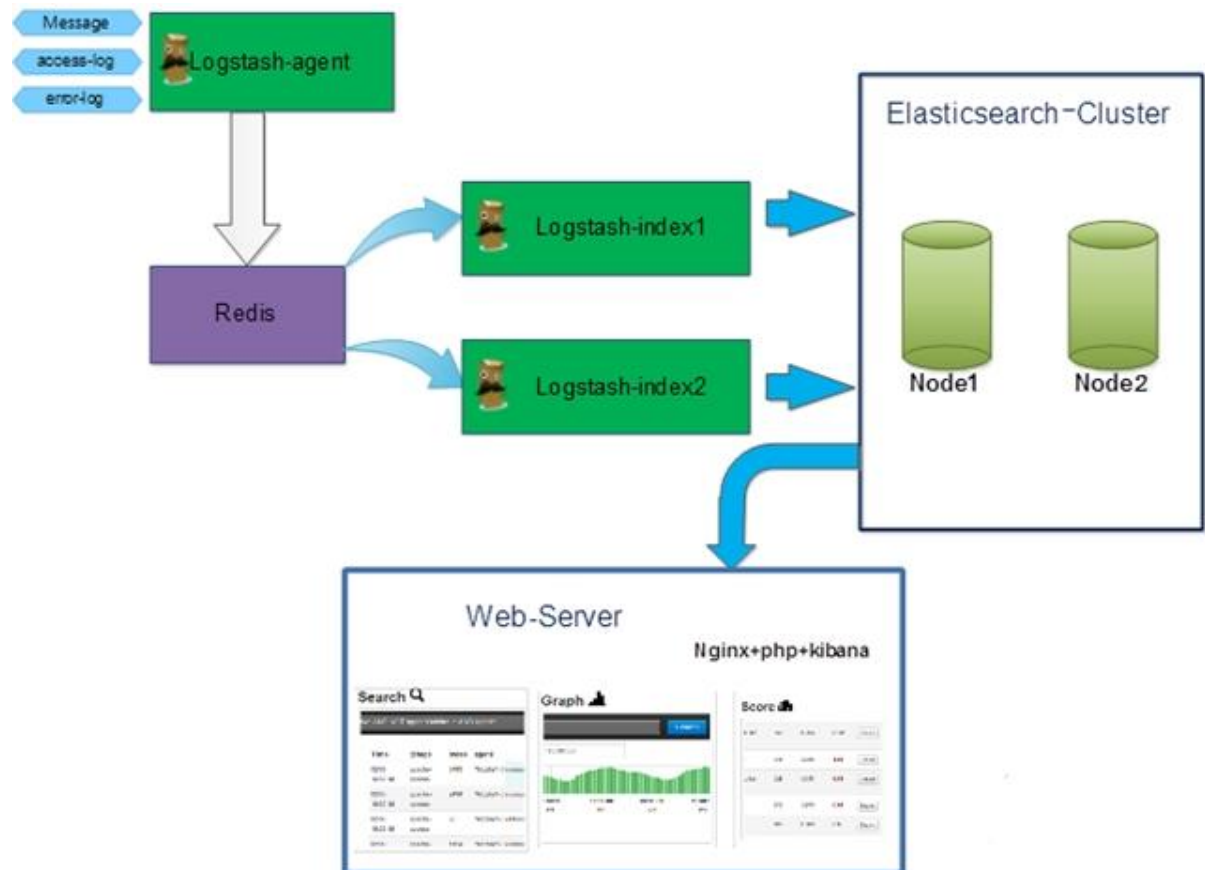
Elasticsearch: 存储各类日志;

Kibana: web 化接口用作查寻和可视化日志;

Logstash Forwarder: logstash client 端用来通过 lumberjack 网络协议发送日志到 logstash server;

#3.ELK 工作流程

在需要收集日志的所有服务上部署 logstash，作为 logstash agent (logstash shipper) 用于监控并过滤收集日志，将过滤后的内容发送到 Redis，然后 logstash indexer 将日志收集在一起交给全文搜索服务 Elasticsearch，可以用 Elasticsearch 进行自定义搜索通过 Kibana 来结合自定义搜索进行页面展示。



我这里为了方便全部安装在一台机子上，也没有用 redis

2.安装 jdk1.8

#因为 Elasticsearch 和 Logstash 要求用到 jdk1.8 以上版本

打开 <http://www.oracle.com/technetwork/java/javase/downloads/index.html> 网页



| Java SE Development Kit 8u144 | | |
|---|-----------|---|
| You must accept the Oracle Binary Code License Agreement for Java SE to download this software. | | |
| <input checked="" type="radio"/> Accept License Agreement <input type="radio"/> Decline License Agreement | | |
| Product / File Description | File Size | Download |
| Linux ARM 32 Hard Float ABI | 77.89 MB | jdk-8u144-linux-arm32-vfp-hflt.tar.gz |
| Linux ARM 64 Hard Float ABI | 74.83 MB | jdk-8u144-linux-arm64-vfp-hflt.tar.gz |
| Linux x86 | 164.65 MB | jdk-8u144-linux-i586.rpm |
| Linux x86 | 175.44 MB | jdk-8u144-linux-i586.tar.gz |
| Linux x64 | 162.1 MB | jdk-8u144-linux-x64.rpm |
| Linux x64 | 176.92 MB | jdk-8u144-linux-x64.tar.gz |
| Mac OS X | 226.6 MB | jdk-8u144-macosx-x64.dmg |
| Solaris SPARC 64-bit | 139.87 MB | jdk-8u144-solaris-sparcv9.tar.Z |
| Solaris SPARC 64-bit | 99.18 MB | jdk-8u144-solaris-sparcv9.tar.gz |
| Solaris x64 | 140.51 MB | jdk-8u144-solaris-x64.tar.Z |
| Solaris x64 | 96.99 MB | jdk-8u144-solaris-x64.tar.gz |
| Windows x86 | 190.94 MB | jdk-8u144-windows-i586.exe |
| Windows x64 | 197.78 MB | jdk-8u144-windows-x64.exe |

```
mkdir /disk1/tools
cd /disk1/tools/
#上传刚刚下载的 jdk-8u144-linux-x64.rpm
rpm -ih jdk-8u144-linux-x64.rpm
java -version
javac -version
```

```
[root@vm1 tools]# java -version
java version "1.8.0_144"
Java(TM) SE Runtime Environment (build 1.8.0_144-b01)
Java HotSpot(TM) 64-Bit Server VM (build 25.144-b01, mixed mode)
[root@vm1 tools]# javac -version
javac 1.8.0_144
```

#设置环境变量

```
cat >/etc/profile.d/java.sh<<EOF
JAVA_HOME=/usr/java/jdk1.8.0_144
JRE_HOME=/usr/java/jdk1.8.0_144/jre
PATH=\$PATH:\$JAVA_HOME/bin:\$JRE_HOME/bin
CLASSPATH=.\$JAVA_HOME/lib/dt.jar:\$JAVA_HOME/lib/tools.jar:\$JRE_HOME/lib
export JAVA_HOME JRE_HOME PATH CLASSPATH
EOF
cat /etc/profile.d/java.sh
source /etc/profile.d/java.sh
echo $JAVA_HOME
```

3.安装 ELK5.6

#1.ELK 安装 (yum)

#ELK yum 安装文档

<https://www.elastic.co/guide/en/elasticsearch/reference/current/rpm.html>
<https://www.elastic.co/guide/en/logstash/current/installing-logstash.html>
<https://www.elastic.co/guide/en/kibana/current/rpm.html>

```
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
cat>/etc/yum.repos.d/elasticsearch.repo<<EOF
[elasticsearch-5.x]
name=Elasticsearch repository for 5.x packages
baseurl=https://artifacts.elastic.co/packages/5.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
EOF
yum install elasticsearch logstash kibana -y
```

| Package | 架构 | 版本 | 源 | 大小 |
|---|--------|-----------|-------------------|------|
| 正在安装: | | | | |
| elasticsearch | noarch | 5.6.0-1 | elasticsearch-5.x | 32 M |
| kibana | x86_64 | 5.6.0-1 | elasticsearch-5.x | 50 M |
| logstash | noarch | 1:5.6.0-1 | elasticsearch-5.x | 98 M |
| 事务概要 | | | | |
| 安装 3 软件包 | | | | |
| 总下载量: 180 M | | | | |
| 安装大小: 407 M | | | | |
| Downloading packages: | | | | |
| (1/3): elasticsearch-5.6.0.rpm 2% [-] 24 kB/s 4.5 MB 02:05:04 ETA | | | | |

#如果觉得下载很慢，也可以单独下载直接安装，不用安装依赖
#下载页面为 <https://www.elastic.co/cn/downloads>，整个 ELK 都能在里面找到

```
cd /disk1/tools/
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-5.6.0.rpm
wget https://artifacts.elastic.co/downloads/logstash/logstash-5.6.0.rpm
wget https://artifacts.elastic.co/downloads/kibana/kibana-5.6.0-x86_64.rpm
```

#yum install 安装 rpm 比 “rpm -ih” 好处的话就连依赖都帮你列出来

```
yum install elasticsearch-5.6.0.rpm logstash-5.6.0.rpm kibana-5.6.0-x86_64.rpm -y
```

#2.设置环境

```
echo 'export PATH=$PATH:/usr/share/elasticsearch/bin:/usr/share/logstash/bin:/usr/share/kibana/bin'>/etc/profile.d/elk.sh
source /etc/profile.d/elk.sh
```

4.elasticsearch 简单配置及启动

#1.查看安装相关

```
rpm -ql elasticsearch-5.6.0-1.noarch
```

```
rpm -ql elasticsearch-5.6.0-1.noarch|grep 'elasticsearch/bin'
cd /etc/elasticsearch/
cp elasticsearch.yml elasticsearch.yml.orig
```

#2.修改配置

#建立数据目录和日志目录

```
mkdir -p /disk1/elkDate/elasticsearch
mkdir -p /disk1/logs/elasticsearch
chown elasticsearch.elasticsearch -R /disk1/elkDate/elasticsearch
chown elasticsearch.elasticsearch -R /disk1/logs/elasticsearch
```

#修改数据目录和日志目录

```
sed -i '/path.data/s#/path/to/data#/disk1/elkDate/elasticsearch#' elasticsearch.yml
sed -i '/path.data/s/#/' elasticsearch.yml
sed -i '/path.logs/s#/path/to/logs#/disk1/logs/elasticsearch#' elasticsearch.yml
sed -i '/path.logs/s/#/' elasticsearch.yml
grep 'path.' elasticsearch.yml
```

#修改 ip 地址为内网

```
sed -i '/network.host/s/0.1/3.11/' elasticsearch.yml
sed -i '/network.host/s/#/' elasticsearch.yml
grep 'network.host' elasticsearch.yml
```

#避免出现跨域问题

```
cat>>/etc/elasticsearch/elasticsearch.yml<<EOF
#by hua
http.cors.allow-origin: "*"
http.cors.enabled: true
#xpack.security.authc:
#  anonymous:
#    username: elastic
#    roles: superuser
#    authz_exception: true
EOF
tail -10 /etc/elasticsearch/elasticsearch.yml
```

#3.启动

```
systemctl daemon-reload
systemctl enable elasticsearch.service
systemctl start elasticsearch
systemctl status elasticsearch
sleep 70
netstat -altnp|grep 9200
```

#浏览器查看

http://192.168.3.11:9200



5.logstash 简单配置及启动

logstash 对于初学者来说是最容易出问题的，所以一下要开 2 个 SSH，一个是命令操作，一个用看查看日志，要保证不要报错。

#1.查看安装相关

```
rpm -ql logstash-5.6.0-1.noarch | egrep -v "/usr/share"
#查看命令
[root@vm1 logstash]# rpm -ql logstash-5.6.0-1.noarch | grep 'logstash/bin'
/usr/share/logstash/bin/cpdump
/usr/share/logstash/bin/ingest-convert.sh
/usr/share/logstash/bin/logstash
/usr/share/logstash/bin/logstash-plugin
/usr/share/logstash/bin/logstash-plugin.bat
/usr/share/logstash/bin/logstash.bat
/usr/share/logstash/bin/logstash.lib.sh
/usr/share/logstash/bin/ruby
/usr/share/logstash/bin/setup.bat
/usr/share/logstash/bin/system-install
```

#2.修改配置

```
cd /etc/logstash/
cp logstash.yml logstash.yml.orig
#建立相关数据和日志目录
mkdir -p /disk1/elkDate/logstash
mkdir -p /disk1/logs/logstash
chown logstash.logstash -R /disk1/elkDate/logstash
```

```
chown logstash.logstash -R /disk1/logs/logstash
```

#修改数据目录和日志目录

```
sed -i '/^path.data/s#/var/lib/logstash#/disk1/elkDate/logstash#' logstash.yml
```

```
sed -i '/^path.logs/s#/var/log/logstash#/disk1/logs/logstash#' logstash.yml
```

```
egrep '^path.' logstash.yml
```

#下面的配置路径一般不需要修改

```
#sed -i '/path.config/s#/conf.d##' logstash.yml
```

#3.配置 pipeline 文件（只是一个测试配置，获取 messages 信息）

#配置文件也可暂时不建立也行，不影响，默认情况是没有任何配置的，没配置启动服务没意义！

#根据默认配置，pipeline 实例文件默认应放置于/etc/logstash/conf.d 目录，此时目录下无实例文件，

#可根据实际情况新建实例，以处理本机 messages 信息为例，如下

```
cd /etc/logstash/conf.d/
```

```
cat>messages.conf<<EOF
```

```
input {
```

```
  file {
```

```
    path => "/var/log/messages"
```

```
  }
```

```
}
```

```
output {
```

```
  elasticsearch {
```

```
    #hosts => ["192.168.3.11:9200","192.168.3.12:9200"]
```

```
    hosts => ["192.168.3.11:9200"]
```

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

```
  }
```

```
  stdout {
```

```
    # codec => rubydebug
```

```
  }
```

```
}
```

```
EOF
```

```
cat messages.conf
```

#从上面知道建立的索引是“messages-年.月.日”的格式，到后面安装后 kibana，web 登陆可以用到

#4.测试

#输入下面命令就会自动启动 logstash，当退出就会停止，如果要长期运行就启动服务的方式。

```
logstash -e 'input { stdin { } } output { stdout { } }'
```

发现报错：

```
[root@vml logstash]# logstash -e 'input { stdin { } } output { stdout { } }'
2017-09-14 18:44:40,913 main ERROR Unable to locate appender "${sys:ls.log.format}_rolling" for logger config "root"
WARNING: Could not find logstash.yml which is typically located in $LS_HOME/config or /etc/logstash. You can specify the p
ath using --path.settings. Continuing using the defaults
Could not find log4j2 configuration at path //usr/share/logstash/config/log4j2.properties. Using default config which logs
errors to the console
The stdin plugin is now waiting for input:
```

解决：

```
ctrl+c 退出
```

```
ln -s /etc/logstash /usr/share/logstash/config
chown logstash.logstash -R /etc/logstash
```

#再次运行，输入 hello world 测试一下

```
[root@vm1 logstash]# logstash -e 'input { stdin { } } output { stdout { } }'
2017-09-14 18:47:42,450 main ERROR Unable to locate appender "${sys:ls.log.format}.rolling" for logger config "root"
Sending Logstash's logs to /disk1/logs/logstash which is now configured via log4j2.properties
The stdin plugin is now waiting for input:
hello world
2017-09-14T10:47:50.229Z 0.0.0.0 hello world
```

上面的错误不理它

#5.启动

#上面配置了一个 messages 日志，为了长期运行，所以启动服务，到后面可以查一下情况

```
systemctl daemon-reload
systemctl enable logstash
systemctl start logstash
systemctl status logstash
```

#要查看 logstash 插件,可以看下面我链接

<https://github.com/logstash-plugins>

6.kibana 简单配置及启动

#1.查看安装相关

```
rpm -ql kibana-5.6.0-1.x86_64 | grep 'kibana/bin'
```

#2.修改配置

```
cd /etc/kibana/
cp kibana.yml kibana.yml.orig
```

#修改 ip 地址为内网网卡 IP

```
sed -i '/#server.host/s/"localhost"/192.168.3.11/' kibana.yml
sed -i '/#server.host/s/#//' kibana.yml
grep 'server.host' kibana.yml
sed -i '/elasticsearch.url/s/localhost/192.168.3.11/' kibana.yml
sed -i '/elasticsearch.url/s/#//' kibana.yml
grep 'elasticsearch.url' kibana.yml
```

#3.启动

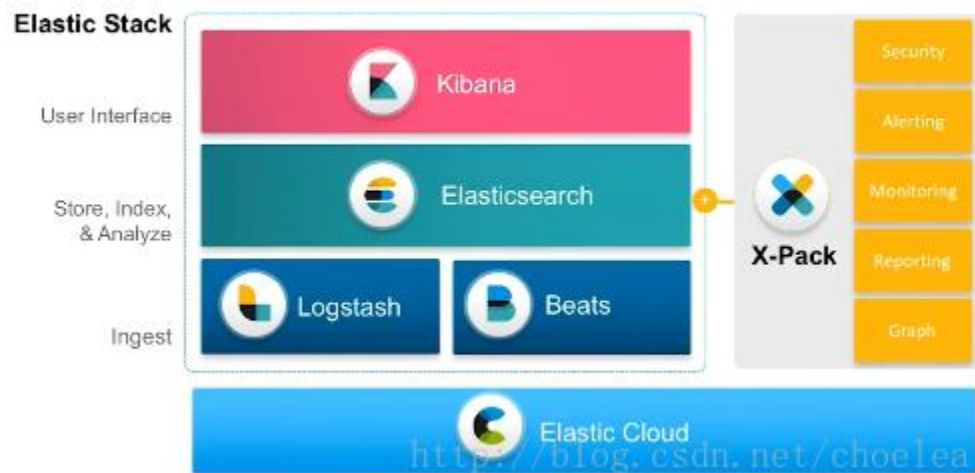
```
systemctl start kibana
systemctl restart kibana
systemctl status kibana
sleep 30
netstat -altnp|grep 5601
```

7. x-pack 插件

官网: <https://www.elastic.co/guide/en/x-pack/current/installing-xpack.html>

在 5.x 版本中一些 es 插件(Shield, Watcher, Marvel, Graph, Reporting)都集成在 x-pack 组件中 Elastic 做了一个 package , 对原本的 marvel、watch、alert 做了一个封装, 形成了 x-pack 。

Introducing the Elastic Stack, X-Pack, and Cloud



#1.给 elk 都安装上 x-pack 插件

#提示都用 y 不要用 N

```
elasticsearch-plugin install x-pack
```

#这个时间比较久

```
kibana-plugin install x-pack
```

#logstash 安装 x-pack 需要安装上面的 2 个, 也可以不安装, 用到的时候再安装

```
logstash-plugin install x-pack
```

#重启

```
systemctl restart elasticsearch
```

```
systemctl status elasticsearch
```

#操作过程中重启不启动不起来, 需要手工启动

```
systemctl restart logstash
```

```
systemctl restart kibana
```

```
sleep 70
```

```
netstat -alntp|grep -i listen
```

#2.修改 logstash 配置

```
cat>>/etc/logstash/logstash.yml <<EOF
```

```
#by hua
```

```
xpack.monitoring.elasticsearch.url: "http://192.168.3.11:9200"
```

```
#xpack.monitoring.elasticsearch.username: "logstash_system"
```

```
#xpack.monitoring.elasticsearch.password: "changeme"
```

```
EOF
```

```
tail -5 /etc/logstash/logstash.yml
```

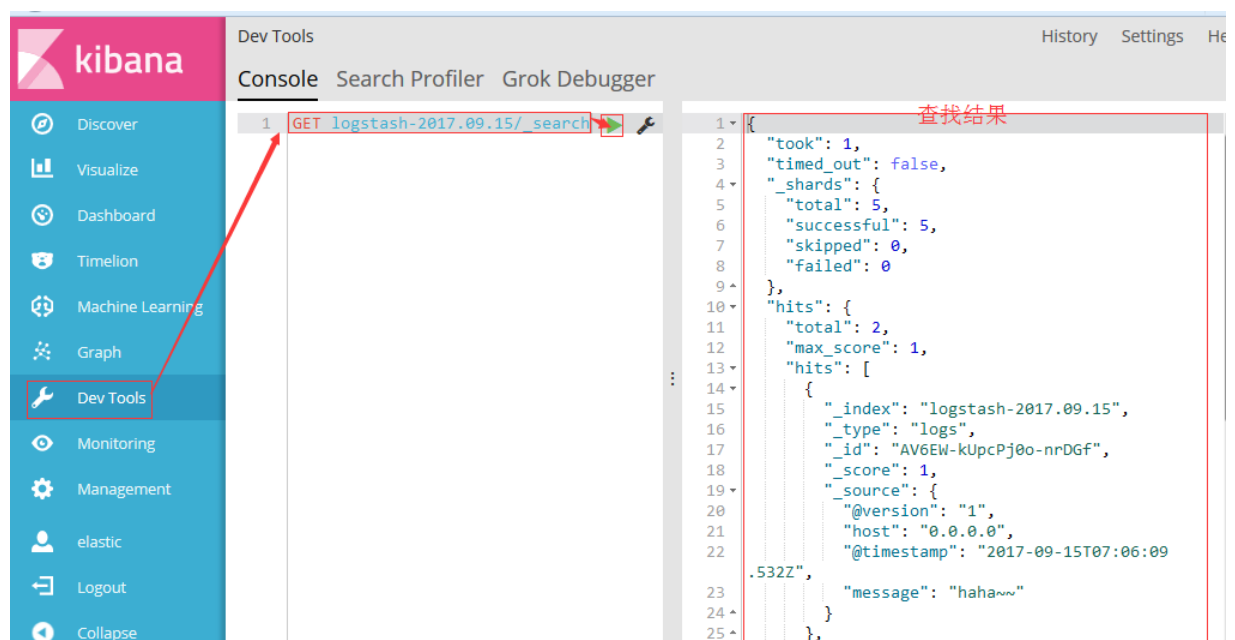
#3.为 logstash 配置添加用户名和密码(为测试添加密码)

因为安装了 x-pack, 所以需要用户名和密码, 如果不添加看日志会报 401 错误, 如果还是 401 说明用户名或密码不正确

```
cd /etc/logstash/conf.d/
>messages.conf
cat>messages.conf<<EOF
input {
  file {
    path => "/var/log/messages"
  }
}
output {
  elasticsearch {
    #hosts => ["192.168.3.11:9200","192.168.3.12:9200"]
    hosts => ["192.168.3.11:9200"]
    user => "elastic"
    password => "123456"
    index => "messages-%{+YYYY.MM.dd}"
  }
  stdout {
    # codec => rubydebug
  }
}
EOF
cat messages.conf
```

#注: 上面到用户默认密码我改为了 123456, 到后面会讲到

配置修改完了, 可以用浏览器登陆 kibana, 在"Dev Tools", 如果看不懂没关系, 后面会讲到今天是 2017.09.15, 就查一下今天的索引信息



The screenshot shows the Kibana interface with the 'Dev Tools' sidebar on the left. The 'Console' tab is active, displaying a search query: `GET logstash-2017.09.15/_search`. The search results are shown on the right, with a red box highlighting the first hit. The results include metadata like `_index`, `_type`, `_id`, `_score`, and `_source`, as well as the actual log message: `"message": "haha~~"`. A red arrow points from the 'Dev Tools' menu item to the console input field.

查找结果

#4.配置 x-pack

官网:

<https://www.elastic.co/guide/en/elasticsearch/reference/current/settings-xpack.html#settings-xpack>

#这里我没有做任何配置，需要的可以看一下下面列表，点击进去查看一下

[Machine Learning Settings in Elasticsearch](#) »

Configuring X-Pack

You configure settings for X-Pack features in the `elasticsearch.yml`, `kibana.yml`, and `logstash.yml` configuration files.

| X-Pack Feature | Elasticsearch Settings | Kibana Settings | Logstash Settings |
|------------------|------------------------|---------------------|---------------------|
| Watcher | Yes | No | No |
| Graph | No | Yes | No |
| Machine learning | Yes | Yes | No |
| Monitoring | Yes | Yes | Yes |
| Reporting | No | Yes | No |
| Security | Yes | Yes | No |

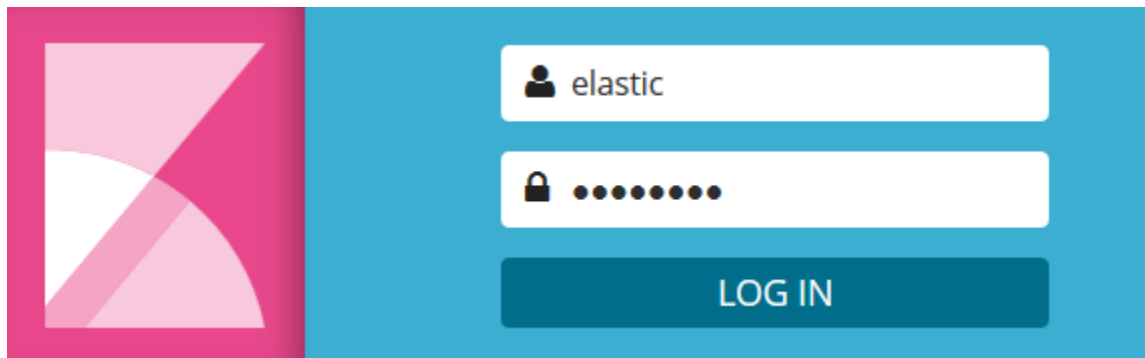
#5.登陆认识一下界面

用浏览器打开 192.168.3.11:5601

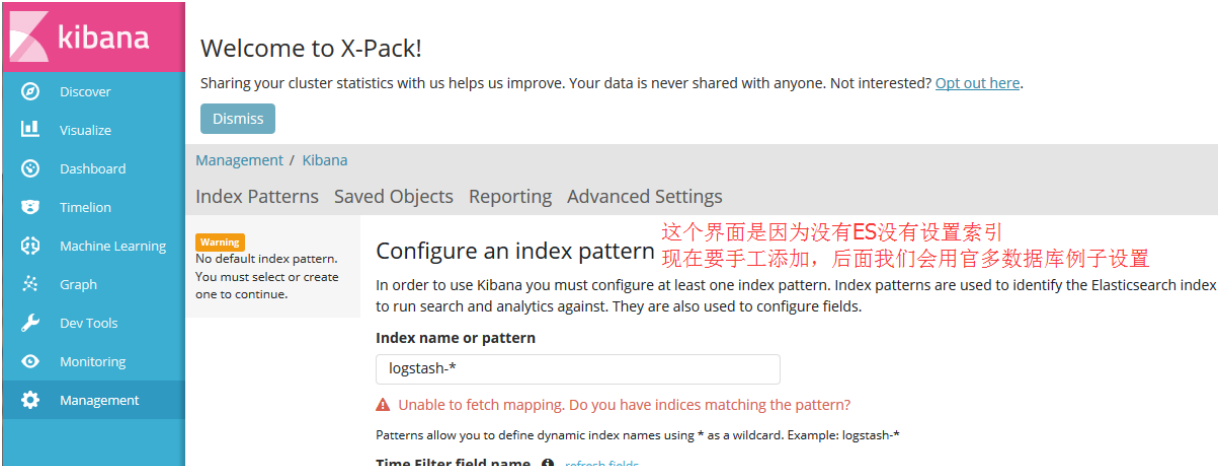
用户名: elastic

密码: changeme

如果打开 Firefox 浏览器输入上面 IP 地址,提示有“检测到该服务器正在将此地址的请求循环重定向”
一般情况是 kibana 配置中 `elasticsearch.url` 的 ip 地址没有配置对



#登陆界面



The screenshot shows the Kibana 'Welcome to X-Pack!' page. On the left is a sidebar with navigation links: Discover, Visualize, Dashboard, Timelion, Machine Learning, Graph, Dev Tools, Monitoring, and Management (highlighted). The main content area has a 'Welcome to X-Pack!' message with a 'Dismiss' button. Below this is a breadcrumb 'Management / Kibana' and a sub-header 'Index Patterns Saved Objects Reporting Advanced Settings'. A warning box states: 'Warning: No default index pattern. You must select or create one to continue.' The main section is titled 'Configure an index pattern' and explains that at least one index pattern must be configured. It includes a text input field for 'Index name or pattern' containing 'logstash-*'. Below the field is a red error message: 'Unable to fetch mapping. Do you have indices matching the pattern?'. Further down, it says 'Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*' and provides a 'Time Filter field name' dropdown with 'refresh fields' selected.

Welcome to X-Pack!

Sharing your cluster statistics with us helps us improve. Your data is never shared with anyone. Not interested? [Opt out here.](#)

Dismiss

Management / Kibana

Index Patterns Saved Objects Reporting Advanced Settings

Warning
No default index pattern.
You must select or create one to continue.

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

Index name or pattern

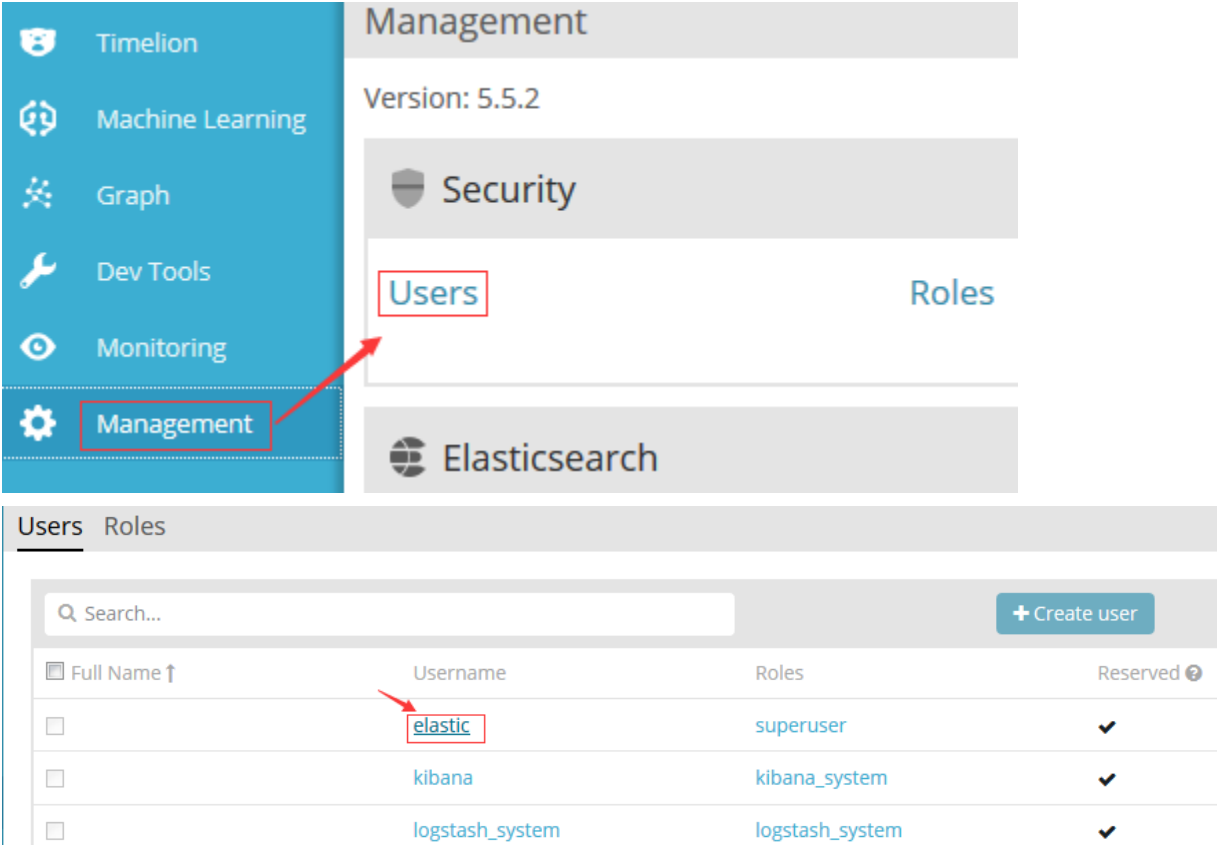
logstash-*

Unable to fetch mapping. Do you have indices matching the pattern?

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

Time Filter field name [refresh fields](#)

#修改密码



The screenshot shows the 'Management' section of Kibana, specifically the 'Users' tab. The sidebar on the left has 'Management' highlighted. The main content area shows 'Management' with 'Version: 5.5.2'. Below this are sections for 'Security' and 'Elasticsearch'. The 'Security' section has 'Users' and 'Roles' tabs, with 'Users' selected. A red box highlights the 'Users' tab, and a red arrow points from the 'Management' tab in the sidebar to it. Below the tabs is a table of users. The table has columns: 'Full Name ↑', 'Username', 'Roles', and 'Reserved ?'. There are three users listed: 'elastic' (superuser), 'kibana' (kibana_system), and 'logstash_system' (logstash_system). A red box highlights the 'elastic' username, and a red arrow points from the 'elastic' username to the 'Reserved ?' column.

Management

Version: 5.5.2

Security

Users Roles

Elasticsearch

Users Roles

Search... [+ Create user](#)

| Full Name ↑ | Username | Roles | Reserved ? |
|--------------------------|-----------------|-----------------|------------|
| <input type="checkbox"/> | elastic | superuser | ✓ |
| <input type="checkbox"/> | kibana | kibana_system | ✓ |
| <input type="checkbox"/> | logstash_system | logstash_system | ✓ |

#从上图可以看到也可以增加用户，这里我就不操作了，太简单了

Users Roles

"elastic" User

Username

elastic

Roles

superuser

Password

[Change Password](#)

Password

Current Password

.....

New Password

把密码改为复杂的，我这里为了方便改为123456

.....

New Password Again, Please

.....

Save Cancel

#开发工具 Dev Tools（对 ES 中 API 操作都在这里）

原先的交互式控制台 Sense，使用户方便的通过浏览器直接与 Elasticsearch 进行交互。从 Kibana 5 开始改名并直接内建在 Kibana，就是 Dev Tools 选项。



二、测试

1.导入官方数据

官网：<https://www.elastic.co/guide/en/kibana/5.6/tutorial-load-dataset.html>

Loading Sample Data

The tutorials in this section rely on the following data sets:

- The complete works of William Shakespeare, suitably parsed into fields. Download this data set by clicking here: [shakespeare.json](#).
- A set of fictitious accounts with randomly generated data. Download this data set by clicking here: [accounts.zip](#)
- A set of randomly generated log files. Download this data set by clicking here: [logs.jsonl.gz](#)

Two of the data sets are compressed. Use the following commands to extract the files:

下载是官网的下载地址:

<https://download.elastic.co/demos/kibana/gettingstarted/shakespeare.json>

<https://download.elastic.co/demos/kibana/gettingstarted/accounts.zip>

<https://download.elastic.co/demos/kibana/gettingstarted/logs.jsonl.gz>

下载上面的例子数据并解压上传到 /disk1/tools/ 中

cd /disk1/tools/

因最版本 5 要用到用户名和密码, 所以你需要 “curl -u 用户名:密码” 还有指定头部分格式为 json

否则会报错, 向 ES 导入数据, 命令如下:

```
curl -u elastic:123456 -H 'Content-Type: application/x-ndjson' \  
-XPOST '192.168.3.11:9200/bank/account/_bulk?pretty' --data-binary @accounts.json
```

```
curl -u elastic:123456 -H 'Content-Type: application/x-ndjson' \  
-XPOST '192.168.3.11:9200/shakespeare/_bulk?pretty' --data-binary @shakespeare.json
```

```
curl -u elastic:123456 -H 'Content-Type: application/x-ndjson' \  
-XPOST '192.168.3.11:9200/_bulk?pretty' --data-binary @logs.jsonl
```

查看所有索引

```
curl -u elastic:123456 '192.168.3.11:9200/_cat/indices?v'
```

结果如下:

```
[root@vm1 tools]# curl -u elastic:123456 '192.168.3.11:9200/_cat/indices?v'
```

| health | status | index | uuid | pri | rep | docs.count | docs.deleted | store.size | pri.store.size |
|--------|--------|---------------------------------|------------------------|-----|-----|------------|--------------|------------|----------------|
| yellow | open | .monitoring-es-6-2017.09.15 | RIGF_89JTiiP3X6EGx1oRg | 1 | 1 | 1521 | 81 | 1mb | 1mb |
| yellow | open | .watcher-history-6-2017.09.14 | B4Jagg1PTw-FMo4IB_Dbyg | 1 | 1 | 110 | 0 | 151.4kb | 151.4kb |
| yellow | open | messages-2017.09.14 | CTBb4XD6RLyMwHEuNR6ZV0 | 5 | 1 | 3 | 0 | 15.2kb | 15.2kb |
| yellow | open | logstash-2015.05.20 | fyk4k5rrQ708t-xgF3qyQA | 5 | 1 | 4750 | 0 | 27.6mb | 27.6mb |
| yellow | open | logstash-2015.05.19 | XBg0PnqbQPyz3-wJ62I6kg | 5 | 1 | 4624 | 0 | 27.7mb | 27.7mb |
| green | open | .security | xMbAlrzrQP2Kw1vysbPt8w | 1 | 0 | 1 | 0 | 2.8kb | 2.8kb |
| yellow | open | .monitoring-kibana-6-2017.09.15 | ASmTvzuXQH0k4YLaRDUENG | 1 | 1 | 75 | 0 | 60.8kb | 60.8kb |
| yellow | open | shakespeare | 2yLPWceMThaWiipbTbBqwQ | 5 | 1 | 108355 | 0 | 28.3mb | 28.3mb |
| yellow | open | .monitoring-alerts-6 | A8M4Mwo8TjqISABq3_ojcw | 1 | 1 | 1 | 0 | 12.4kb | 12.4kb |
| yellow | open | .monitoring-es-6-2017.09.14 | fedpvJdFSo-lPKZoYgIj6A | 1 | 1 | 1859 | 75 | 1.3mb | 1.3mb |
| yellow | open | bank | N8KwfwEnTt0hCxPzkkxCjg | 5 | 1 | 1000 | 0 | 640.3kb | 640.3kb |
| yellow | open | .watcher-history-6-2017.09.15 | fZrEtGivTiWnmUeAuhytsw | 1 | 1 | 70 | 0 | 224.4kb | 224.4kb |
| yellow | open | .triggered_watches | 0YkI1MqsT76mYJyzk0Qxcw | 1 | 1 | 0 | 0 | 10.4kb | 10.4kb |
| yellow | open | .kibana | VEj1CGL0S32V78j4Dt1hBg | 1 | 1 | 1 | 0 | 3.8kb | 3.8kb |
| yellow | open | .monitoring-kibana-6-2017.09.14 | 6Gx4rZB3QryQc9D3nUZevg | 1 | 1 | 135 | 0 | 204.7kb | 204.7kb |
| yellow | open | logstash-2015.05.18 | rTCU4jdQTy-bVeAYi_W1QA | 5 | 1 | 4631 | 0 | 26.7mb | 26.7mb |
| yellow | open | .watches | HH-CFB0rTQa0hs0DbS0L-Q | 1 | 1 | 4 | 0 | 30.9kb | 30.9kb |

有了上面的数据我们就可以添加索引了

2. 在 kibana 添加设置数据索引

打开浏览器输入: 192.168.3.11:5601 并输入用户名和密码登陆 kibana

#1. 添加有日期索引

如我们按上面添加一个以 logstash 开头的索引, 这是一个普通的日志, 是带日期的

Management / Kibana

Index PatternsSaved ObjectsReportingAdvanced Settings

Warning

No default index pattern.
You must select or create one to continue.

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

Index name or pattern

logstash-* *号是通配符表示所有

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

Time Filter field name refresh fields

utc_time 选择一个日期

☐ Expand index pattern when searching [DEPRECATED]

With this option selected, searches against any time-based index pattern that contains a wildcard will automatically be expanded to query only the indices that contain data within the currently selected time range.

Searching against the index pattern *logstash-** will actually query Elasticsearch for the specific matching indices (e.g. *logstash-2015.12.21*) that fall within the current time range.

With recent changes to Elasticsearch, this option should no longer be necessary and will likely be removed in future versions of Kibana.

☐ Use event times to create index names [DEPRECATED]

Create

Management / Kibana

Index PatternsSaved ObjectsReportingAdvanced Settings

+ Create Index Pattern

★ logstash-*

创建索引成功后的图

★ logstash-*

Time Filter field name: utc_time

This page lists every field in the **logstash-*** index and the field's associated core type as recorded by Elasticsearch. While this list allows you to view the core type of each field, changing field types must be done using Elasticsearch's [Mapping API](#)

fields (87)

scripted fields (0)

source filters (0)

Filter

All field types

| name | type | format | searchable | aggregatable | excluded | controls |
|------------------|--------|--------|------------|--------------|----------|----------|
| @message | string | | ✓ | | | |
| @message.keyword | string | | ✓ | ✓ | | |
| @tags | string | | ✓ | | | |

#2.创建普通索引

点上图中的

+ Create Index Pattern

，创建一个新索引，这此选择“bank”

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

Index name or pattern

bank

这个是没有日期的，自动识别了
没有时间字段让我们选择

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

Time Filter field name [refresh fields](#)

The indices which match this index pattern don't contain any time fields. ▼

Create

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

Index name or pattern

bank

当输入索引名字时自动识别有哪些选项
这里没有时间字段所以没有此选项

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

Time Filter field name [refresh fields](#)

The indices which match this index pattern don't contain any time fields. ▼

Create

Management / Kibana

Index Patterns Saved Objects Reporting Advanced Settings

+ Create Index Pattern

★ logstash-*

bank

bank

This page lists every field in the **bank** index and the field's associated core type as recorded by Elasticsearch. While this list allows you to view the core type of each field, changing field types must be done using Elasticsearch's [Mapping API](#).

fields (24) scripted fields (0) source filters (0)

Q Filter 创建后的结果 All field types ▼

| name | type | format | searchable | aggregatable | excluded | controls |
|--------|--------|--------|------------|--------------|----------|----------|
| _id | string | | ✓ | | | |
| _index | string | | ✓ | ✓ | | |
| _score | number | | | | | |

三、Elasticsearch RESTful API 操作（文档）

ES API 官网: <https://www.elastic.co/guide/en/elasticsearch/reference/current/docs.html>

RESTful API 是用 kibana 的 Dev Tools 做 CURD 操作的

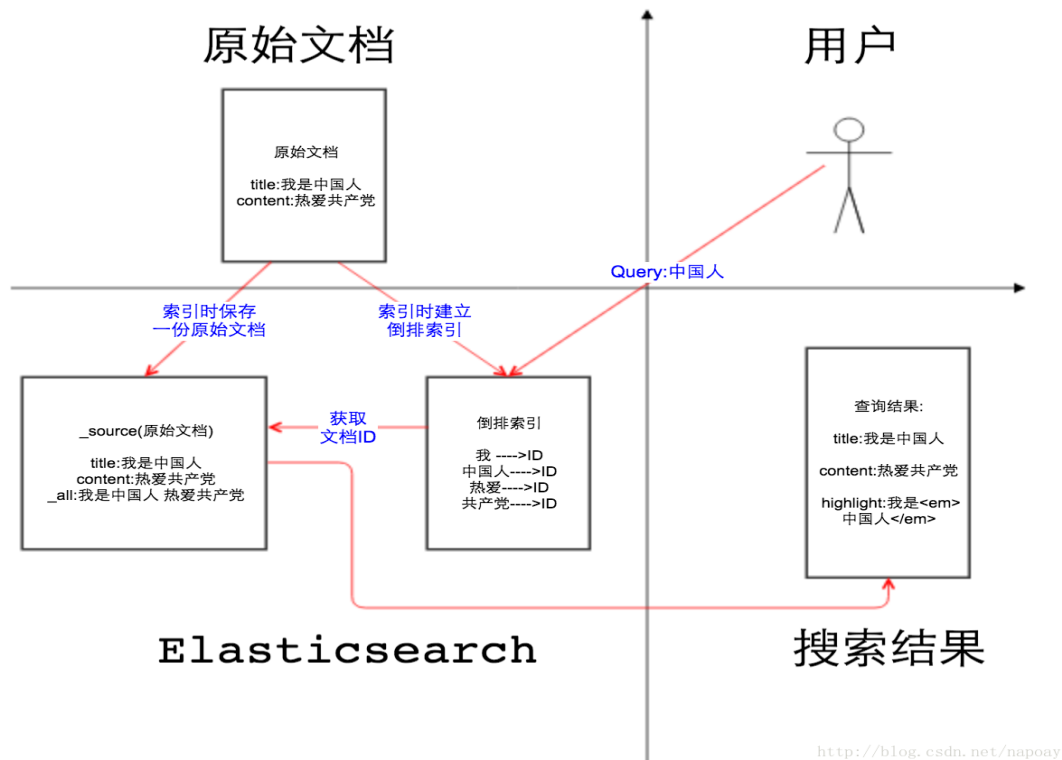
打开浏览器输入 <http://192.168.3.11:5601> 登陆 kibana，找到“Dev Tools”

0. 相关知识

#1. Elasticsearch 中的 _source、_all、store 和 index 属性

Elasticsearch 中有几个关键属性容易混淆，很多人搞不清楚 _source 字段里存储的是什么？store 属性

的 true 或 false 和 _source 字段有什么关系？store 属性设置为 true 和 _all 有什么关系？index 属性又起到什么作用？什么时候设置 store 属性为 true？什么时候应该开启 _all 字段？本文通过图解的方式，深入理解 Elasticsearch 中的 _source、_all、store 和 index 属性。



具体详见：<http://blog.csdn.net/napoay/article/details/62233031>

Elasticsearch 的内置字段以及类型如下：

| | |
|------|--|
| 内置字段 | _uid, _id, _type, _source, _all, _analyzer, _boost, _parent, _routing, _index, _size, _timestamp, _ttl |
| 字段类型 | String, Integer/long, Float/double, Boolean, Null, Date |

1.用 put 创建索引

官方：https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-index_.html

中文链接：<http://cwiki.apachecn.org/pages/viewpage.action?pageId=4882789>

创建索引格式是：PUT <索引名字>/<type 名字>/[ID]

ID 可以不设置会随机生成， POST <索引名字>/<type 名字>

例子 1.

创建一个关于学生成绩的索引，索引名是 stu，类型是 cj(成绩简写)

#格式如下：输入完之后点绿色三角就执行了

```
PUT stu/cj/1
{
  "sid":1,
  "name":"刘一",
```

```
"yuwen":78,
"shuxue":81
}
```

Dev Tools

Console Search Profiler Grok Debugger

1. 编写

```
1 PUT stu/cj/1
2 {
3   "sid":1,
4   "name":"刘一",
5   "yuwen":78,
6   "shuxue":81
7 }
```

2. 运行

3. 结果

```
1 {
2   "_index": "stu",
3   "_type": "cj",
4   "_id": "1",
5   "_version": 1,
6   "result": "created",
7   "_shards": {
8     "total": 2,
9     "successful": 1,
10    "failed": 0
11  },
12   "created": true
13 }
```

从执行的结果知：
stu就是索引名，cj就是type，1就是id

同理我们也建立多几个

```
PUT stu/cj/2
```

```
{
  "sid":2,
  "name":"陈二",
  "yuwen":81,
  "shuxue":76
}
```

```
PUT stu/cj/3
```

```
{
  "sid":3,
  "name":"张三",
  "yuwen":55,
  "shuxue":59
}
```

```
PUT stu/cj/4
```

```
{
  "sid":4,
  "name":"李四",
  "yuwen":56,
  "shuxue":61
}
```

```
PUT stu/cj/5
```

```
{
  "sid":5,
  "name":"王五",

```

```
"yuwen":69,  
"shuxue":77  
}
```

Dev Tools

Console Search Profiler Grok Debugger

```
7 ^ }  
8  
9 PUT stu/cj/2 选中可以批量执行 ▶  
10 {  
11   "sid":2,  
12   "name":"陈二",  
13   "yuwen":81,  
14   "shuxue":76  
15 ^ }  
16  
17 PUT stu/cj/3  
18 {  
19   "sid":3,  
20   "name":"张三",  
21   "yuwen":55,  
22   "shuxue":59  
23 ^ }  
24  
25 PUT stu/cj/4  
26 {  
27   "sid":4,  
28   "name":"李四",  
29   "yuwen":56,  
30   "shuxue":61  
31 ^ }  
32  
33 PUT stu/cj/5  
34 {  
35   "sid":5,  
36   "name":"王五",  
37   "yuwen":69,  
38   "shuxue":77  
39 ^ }
```

```
1 # PUT stu/cj/2  
2 { }  
15  
16 # PUT stu/cj/3  
17 { }  
30  
31 # PUT stu/cj/4  
32 { }  
45  
46 # PUT stu/cj/5  
47 {  
48   "_index": "stu",  
49   "_type": "cj",  
50   "_id": "5",  
51   "_version": 1,  
52   "result": "created",  
53   "_shards": {  
54     "total": 2,  
55     "successful": 1,  
56     "failed": 0  
57 ^ },  
58   "created": true  
59 ^ }
```

例子 2

创建一个关于学生成绩的索引，索引名是 `stu`，类型是 `lx`(联系方式)

```
PUT stu/lx/1  
{  
  "sid":1,  
  "name":"刘一",  
  "addr":"幸福小区 1",  
  "tel":"136123456781"  
}  
  
PUT stu/lx/2  
{  
  "sid":2,  
  "name":"陈二",  
  "addr":"快乐小区 2",  
  "tel":"130123456782"
```



```

}

PUT stu/lx/3
{
  "sid":3,
  "name":"张三",
  "addr":"幸福小区 3",
  "tel":"138123456783"
}

PUT stu/lx/4
{
  "sid":4,
  "name":"李四",
  "addr":"幸福小区 4",
  "tel":"136123456784"
}

PUT stu/lx/5
{
  "sid":5,
  "name":"王五",
  "addr":"快乐小区 5",
  "tel":"138123456785"
}

```

#创建索引的其它选项

#1) 防止重复 ID (op_type=create)

当我们创建一个新索引的时候，如果 ID 相同则会更新掉，能不能在之前做一个判断，如果 ID 存在则报错不让执行，这时可以使用“op_type=create”

例子：在学生成绩中插入一个序列为 5 的

```

PUT stu/cj/5?op_type=create
{
  "sid":5,
  "name":"测试",
  "yuwen":10,
  "shuxue":11
}

```

```
PUT stu/cj/5?op_type=create
{
  "sid":5,
  "name":"测试",
  "yuwen":10,
  "shuxue":11
}
```

```
1 {
2   "error": {
3     "root_cause": [  因为id为5已经存在了，所以报错不让插入
4       {
5         "type": "version_conflict_engine_exception",
6         "reason": "[cj][5]: version conflict, document already exists (current
7           version [1]),
8           "index_uuid": "uNrFgREGTo-wbsE4ec7-1w",
9           "shard": "1",
10          "index": "stu"
11        },
12        "type": "version_conflict_engine_exception",
13        "reason": "[cj][5]: version conflict, document already exists (current
14          version [1]),
15          "index_uuid": "uNrFgREGTo-wbsE4ec7-1w",
16          "shard": "1",
17          "index": "stu"
18        },
19        "status": 409
20      }
21    }
```

#2) 设置超时时间(timeout)

语法 **PUT <索引>/<类型>/<ID>? timeout=n**

单位默认是秒可以是 s m h 分别是秒分时

```
PUT stu/cj/66?timeout=5s
```

```
{
  "sid":66,
  "name":"柳柳",
  "yuwen":81,
  "shuxue":89
}
```

2.get 获取文档信息

#1.简单获取

官网: <https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-get.html>

一般语法: **GET <索引>/<type>/<ID>**

就上面我们刚刚建立的索引的例子

例子: 查看学生成绩中 id 分别为 123 的学生成绩所有信息

命令:

```
GET stu/cj/1
GET stu/cj/2
GET stu/cj/3
```

#获取索引初始化设置

```
GET stu/_settings
```

#2.获取多个文档 (_mget)

官网: <https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-multi-get.html>

多 GET API 允许基于索引, 类型 (可选) 和 ID (也可能路由) 获取多个文档用

上面的例子也可以写成下面的

```
GET stu/cj/_mget
{
  "ids":["1","2","3"]
}
```

```
}
```

#如果想获取 stu 下所有 type 信息为 cj 和 xl 的 ID，如果 ID 都为 1，可以这样写

```
GET stu/_mget/
{
  "docs" : [
    {
      "_type": "cj",
      "_id" : "1"
    },
    {
      "_type": "lx",
      "_id" : "1"
    }
  ]
}
```

#3.获取某 type 下所有 id(_search)

如获取索引为 stu，type 为 cj 的所有 ID

```
GET stu/cj/_search
```

3.Update 更新信息

#1.信息更新 (_update)

例如，我们把名字为“刘一”的语文成绩从 78 更新为 81，

```
POST stu/cj/1/_update
{
  "doc": {
    "yuwen": 81
  }
}
```

注：如果不存在则会变成插入

4. bulk 批量操作（不能美化操作）

官网：<https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html>

Bulk API，能够在一个单一的 API 调用执行多项索引/删除操作。这可以大大提高索引速度。

该 REST API 端点/_bulk，它遵循 JSON 结构：

```
action_and_meta_data\n
optional_source\n
action_and_meta_data\n
optional_source\n
....
action_and_meta_data\n
```

optional_source\n

注意：数据的最终行必须以换行符结束\n。

可能的操作有 `index`, `create`, `delete` 和 `update`, `index` 和 `create` 期望在下一行的作为源, 并与索引 API 有相同的语义。(如果文件具有相同的索引和类型的文件已经存在, 就会创建失败, 必要时候而索引回添加或替换文件)。`delete` 不会作为下一行的源, 并与 `delete` API 中具有相同的语义。`update` 是希望部分文档, `upsert` 和脚本及其选项能够在下一行指定。

| action (行为) | 解释 |
|---------------------|---------------|
| <code>create</code> | 当文档不存在时创建之。 |
| <code>index</code> | 创建新文档或替换已有文档。 |
| <code>update</code> | 局部更新文档。 |
| <code>delete</code> | 删除一个文档。 |

特别注意：不能美化操作如：

POST _bulk

```
{"delete":{"_index": "stu","_type": "cj","_id": "100"}}
```

不能美化写成:

POST _bulk

```
{
  "delete":
  {
    "_index": "stu",
    "_type": "cj",
    "_id": "100"
  }
}
```

例子:

在 `cj` 中添加 `id` 为 6, 删除 `id` 为 1; 创建了一个 `type` 为 `test`, `id` 为 3, 然后进行修改

POST _bulk

```
{ "index" : { "_index" : "stu", "_type" : "cj", "_id" : "6" } }
{ "name" : "赵六", "yuwen":91,"shuxue":100}
{ "delete" : { "_index" : "stu", "_type" : "cj", "_id" : "1" } }
{ "create" : { "_index" : "stu", "_type" : "test", "_id" : "3" } }
{ "name":"测试 1", "yuwen":11,"shuxue":11}
{ "update" : { "_id" : "3", "_type" : "test", "_index" : "stu" } }
{ "doc" : { "name" : "修改 1", "yuwen":100,"shuxue":100} }
```

5.Delete 删除操作

#1.DELETE 简单删除

删除操作比较简单就直接 `DELETE <索引>/<type>/<id>`

如删除 `cj` 中 `id` 为 6 的信息:

```
DELETE stu/cj/6
```

#2. _delete_by_query 查询匹配删除

用法是使用 _delete_by_query 对每个查询匹配的文档执行删除

如删 stu 索引，type 为 test，id 为 3

```
POST stu/test/_delete_by_query
{
  "query":{
    "match":{
      "sid": "3"
    }
  }
}
```

如果想删除 stu 索引所有类型 id 为 3 的可以把 type 去掉变成

```
POST stu/_delete_by_query
{
  "query":{
    "match":{
      "sid": "3"
    }
  }
}
```

这样 type 为 cj 和 lx 的的 sid 为 3 信息都被删除了

四、Elasticsearch RESTful API 操作（search）

1._search 查找

search API 允许你执行一个搜索查询并返回与查询匹配的搜索点击。可以使用简单的查询字符串作为参数或使用请求主体提供查询。

Multi-Index, Multi-Type

所有 search API 可以应用于索引内的跨多个类型，并跨多个索引应用，支持多索引语法。

#1.搜索所有信息

例：查找学生为王五的所有信息，因对中文支持不好，用 name:王五有问题，所以改用 sid

```
GET /stu/_search?q=sid:5
```

#2.在特定类型中进行搜索

例：在 cj 和 lx 也查找 sid 为 5 的

```
GET /stu/cj,lx/_search?q=sid:5
```

我们还可以在多个索引之间搜索具有某个标签的所有 tweet(例如，当每个用户都有自己的索引时)：

```
GET /kimchy,elasticsearch/tweet/_search?q=tag:wow
```

或者我们可以使用 _all 占位符搜索所有可用索引中的所有 tweet：

```
GET /_all/tweet/_search?q=tag:wow
```

甚至搜索所有的索引和所有类型：

默认情况下，`elasticsearch` 拒绝将查询超过 1000 个分片的搜索请求。原因是这样大量的分片使协调节点的工作非常耗费 CPU 和内存。使用较少的较大碎片这种方式组织数据通常是一个更好的主意。如果您想绕过此限制（不鼓励），可以将 `action.search.shard_count.limit` 集群设置更新为更大的值。

#3.可用参数

URI 中允许使用的参数有:

| Name | Description |
|--------------------------|---|
| q | 查询字符串(映射到 <code>query_string</code> 查询，有关更多详细信息，请参阅查询字符串查询) |
| df | 在查询中未定义字段前缀时使用的默认字段。 |
| analyzer | 分析查询字符串时使用的分析器名称。 |
| lowercase_expanded_terms | 应将条款自动缩小或不缩小。默认为 <code>true</code> 。 |
| analyze_wildcard | 应该分析通配符和前缀查询还是不分析。默认为 <code>false</code> 。 |
| default_operator | 要使用的默认运算符，可以是 <code>AND</code> 或 <code>OR</code> 。默认为 <code>OR</code> 。 |
| lenient | 如果设置为 <code>true</code> 将导致基于格式的失败（例如向数字字段提供文本）被忽略。默认为 <code>false</code> 。 |
| explain | 对于每个命中，包含对如何计算命中的计分的解释。 |
| _source | 设置为 <code>false</code> 以禁用检索 <code>_source</code> 字段。您还可以使用 <code>_source_include&_source_exclude</code> 检索文档的一部分（有关更多详细信息，请参阅请求主体文档）。 |
| stored_fields | 为每次命中返回文档的选择性存储字段，逗号分隔。未指定任何值将不会返回任何字段。 |
| sort | 排序执行。可以是 <code>fieldName</code> 或 <code>fieldName: asc/fieldName: desc</code> 的形式。 <code>fieldName</code> 可以是文档中的实际字段，也可以是指示基于分数排序的特殊 <code>_score</code> 名称。可以有几个 <code>sort</code> 参数（顺序很重要）。 |
| track_scores | 排序时，设置为 <code>true</code> 以便仍然跟踪分数并将其作为每次匹配的一部分返回。 |
| timeout | 搜索超时，将搜索请求限制为在指定的时间值内执行并且保留与到期时累积的点击数。默认为无超时。 |
| terminate_after | 要为每个分片收集的文档的最大数量，到达时，查询执行将提前终止。如果设置，响应将有布尔型字段 <code>terminated_early</code> 以指示查询执行是否实际已提前终止。默认为无 <code>terminate_after</code> 。 |

| Name | Description |
|-------------|--|
| from | 从命中的索引开始返回。默认值为 0。 |
| size | 要返回的匹配数。默认值为 10。 |
| search_type | 要执行的搜索操作的类型。可以是 dfs_query_then_fetch 或 query_then_fetch。默认为 query_then_fetch。有关可以执行的不同类型搜索的更多详细信息，请参阅搜索类型。 |

2.通过 URI Search 查找

#1.测过 URI search 查找

URI 只是把上面中最前面的/去掉，如

```
GET /stu/_search?q=sid:5
```

改为

```
GET stu/_search?q=sid:5
```

#2.常用参数

URI 中允许使用的参数有:

| Name | Description |
|--------------------------|---|
| q | 查询字符串(映射到 query_string 查询，有关更多详细信息，请参阅查询字符串查询) |
| df | 在查询中未定义字段前缀时使用的默认字段。 |
| analyzer | 分析查询字符串时使用的分析器名称。 |
| lowercase_expanded_terms | 应将条款自动缩小或不缩小。默认为 true。 |
| analyze_wildcard | 应该分析通配符和前缀查询还是不分析。默认为 false。 |
| default_operator | 要使用的默认运算符，可以是 AND 或 OR 。默认为 OR。 |
| lenient | 如果设置为 true 将导致基于格式的失败（例如向数字字段提供文本）被忽略。默认为 false。 |
| explain | 对于每个命中，包含对如何计算命中的计分的解释。 |
| _source | 设置为 false 以禁用检索 _source 字段。您还可以使用 _source_include&_source_exclude 检索文档的一部分（有关更多详细信息，请参阅请求主体文档）。 |
| stored_fields | 为每次命中返回文档的选择性存储字段，逗号分隔。未指定任何值将不会返回任何字段。 |
| sort | 排序执行。可以是 fieldName 或 fieldName: asc/fieldName: desc |

| Name | Description |
|-----------------|--|
| | 的形式。 fieldName 可以是文档中的实际字段，也可以是指示基于分数排序的特殊 _score 名称。可以有几个 sort 参数（顺序很重要）。 |
| track_scores | 排序时，设置为 true 以便仍然跟踪分数并将其作为每次匹配的一部分返回。 |
| timeout | 搜索超时，将搜索请求限制为在指定的时间值内执行并且保留与到期时累积的点击数。默认为无超时。 |
| terminate_after | 要为每个分片收集的文档的最大数量，到达时，查询执行将提前终止。如果设置，响应将有布尔型字段 terminated_early 以指示查询执行是否实际已提前终止。默认为无 terminate_after。 |
| from | 从命中的索引开始返回。默认值为 0。 |
| size | 要返回的匹配数。默认值为 10。 |
| search_type | 要执行的搜索操作的类型。可以是 dfs_query_then_fetch 或 query_then_fetch。默认为 query_then_fetch。有关可以执行的不同类型搜索的更多详细信息，请参阅搜索类型。 |

3.请求主体查找

太多了，直接给链接看

<https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-body.html>

还有其它查找方式，不一一列出来

五、elasticsearch 查询 DSL

官方：<https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html>

中文：<http://cwiki.apachecn.org/pages/viewpage.action?pageId=4260468>

1.Match ALL 查询

最简单的查询，它匹配所有文档

```
GET /_search
{
  "query": {
    "match_all": {}
  }
}
```

2.全文搜索

高级别的全文搜索通常用于在全文字段（例如：一封邮件的正文）上进行全文搜索。它们了解如何分析查询的字段，并在执行之前将每个字段的分析器（或搜索分析器）应用于查询字符串。

这样的查询有以下这些：

- [匹配查询 \(match query\)](#)
用于执行全文查询的标准查询，包括模糊匹配和词组或邻近程度的查询
- [短语匹配查询 \(match_phrase query\)](#)
与匹配查询类似，但是是用于更加精确的匹配相似的词组或单词。
- [短语前缀匹配查询 \(match_phrase_prefix query\)](#)
这是一种弱类型的查询，类似短语匹配查询，但是它对最终要查询的单词的前缀进行模糊匹配的查询
- [多字段查询 \(multi_match query\)](#)
可以用来对多个字段的版本进行匹配查询
- [常用术语查询 \(common_terms query\)](#)
可以对一些比较专业的偏门词语进行的更加专业的查询
- [查询语句查询 \(query_string query\)](#)
与 lucene 查询语句的语法结合的更加紧密的一种查询，允许你在一个查询语句中使用多个 特殊条件关键字（如：AND|OR|NOT ）对多个字段进行查询，当然这种查询仅限专家用户去使用。
- [简单查询语句 \(simple query string\)](#)
是一种适合直接暴露给用户的简单的且具有非常完善的查询语法的查询语句

我这里只拿几个简单的来讲一下

#1.匹配查询

匹配查询可以接受文本、数字及日期数据，进行分析然后构造查询。

例：查索引 stu 下名字 “name” 为 “王五” 的所有信息

```
GET stu/_search
{
  "query": {
    "match": {
      "name": "王五"
    }
  }
}
```

match-匹配查询

匹配查询的类型为 **boolean**。这意味着分析所提供的文本，并且在分析过程中根据所提供的文本构建一个 **boolean** 查询。运算符标志可以设置为 **or** 或 **and** 控制布尔子句（默认为 **or**）。可以使用 **minimum_should_match** 这个参数来设置要匹配的可选的 **should** 子句的最小数量。

分析器可以设置为控制哪个分析器将对文本执行分析处理。它默认为字段显式映射定义或默认搜索分析器。

lenient 参数可以设置为 **true**，用于忽略由数据类型不匹配引起的异常，例如：尝试使用文本查询字符串查询数字字段。默认为 **false**。

#2.短语查询

match_phrase 查询分析文本，并从分析的文本中创建短语查询。

例：查 stu 的 lx 中，地址包含有 “幸福” 两字的信息

```
GET stu/lx/_search
{
  "query": {
    "match_phrase": {
      "addr": "幸福"
    }
  }
}
```

#3.短语前缀匹配查询

例：查 stu 的 lx 中，手机号包含“136123”开头的所有信息

```
GET stu/lx/_search
{
  "query": {
    "match_phrase_prefix": {
      "tel": "136123"
    }
  }
}
```

#4.查询语句查询

太多了，请查看链接 <http://cwiki.apachecn.org/pages/viewpage.action?pageId=4883355>

3.其它

还有其它 API，如索引 API、cat APIs、Mapping(映射)，这些我都不一一兴趣，上面讲了这么多自己学习应该可以了。

链接：<http://cwiki.apachecn.org/pages/viewpage.action?pageId=4260364>

六、logstash 配置讲解

1.简单的例子

#1.创建一个简单的例子

我现在先把 logstash 停止，我把把 logstash 的测试命令“logstash -e 'input { stdin { } } output { stdout { } }'”，屏幕输出的改为输入 elasticsearch 中，操作如下：

```
cd /etc/logstash/conf.d/
cat>hello.conf<<EOF
input {
  stdin { }
}
output {
  elasticsearch {
```

```

#hosts => ["192.168.3.11:9200","192.168.3.12:9200"]
hosts => ["192.168.3.11:9200"]
user => "elastic"
password => "123456"
}
stdout {
  # codec => rubydebug
}
}
EOF

```

cat hello.conf

logstash -f hello.conf

#有提示输入的情况下输入“hello,my name is hua” 然后回车

#再输入，“nice to meet you” 回车，最后按 ctrl+c 退出

```

[root@vm1 conf.d]# logstash -f hello.conf
Sending Logstash's logs to /disk1/logs/logstash which is now configured via log4j2.properties
The stdin plugin is now waiting for input:
hello,my name is hua
2017-09-15T09:34:49.826Z 0.0.0.0 hello,my name is hua
nice to meet you
2017-09-15T09:35:39.160Z 0.0.0.0 nice to meet you
^C[root@vm1 conf.d]# Ctrl+c

```

#上面结果没有输出到屏幕上而是输入到 ES 中，它的索引默认是 “logstash+今天的日期”

#现在登陆 kibana 的开发工具，执行下面命令

#日期今天的日期，根据各自的情况而定

GET logstash-2017.09.15/_search

1 GET logstash-2017.09.15/_search 执行

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

默认配置 - 按照每日日期建立索引

你将发现 Logstash 可以足够灵巧的在 Elasticsearch 上建立索引... 每天会按照默认格式是 logstash-YYYY.MM.DD 来建立索引。在午夜(GMT)，Logstash 自动按照时间戳更新索引。我们可以根据追溯多长时间的数据作为依据来制定保持多少数据，当然你也可以把比较老的数据迁移到其他的地方(重新索引)来方便查询，此外如果仅仅是简单的删除一段时间数据我们可以使用 Elasticsearch Curator。

#2.配置文件的重要组成部分

从上面的例子来讲，logstash 配置一般分为三个部分

- 输入: `input {}`, 必选
- 输出: `output {}`, 必选
- 过滤: `filter {}`, 可选

一句话: 配置文件必要有一个 **input** 一个 **output** !

2.logstash 配置语法

#1.语法

Logstash 设计了自己的 DSL —— 有点像 Puppet 的 DSL, 或许因为都是用 Ruby 语言写的吧 —— 包括有区域, 注释, 数据类型(布尔值, 字符串, 数值, 数组, 哈希), 条件判断, 字段引用等。

#2.区段

Logstash 用 `{}` 来定义区域。区域内可以包括插件区域定义, 你可以在一个区域内定义多个插件。插件区域内则可以定义键值对设置。示例如下:

```
input {  
  stdin {}  
  syslog {}  
}
```

#3.数据类型

Logstash 支持少量的数据值类型:

- bool

```
debug => true
```

- string

```
host => "hostname"
```

- number

```
port => 514
```

- array

```
match => ["datetime", "UNIX", "ISO8601"]
```

- hash

```
options => {  
  key1 => "value1",  
  key2 => "value2"  
}
```

注意: 如果你用的版本低于 1.2.0, 哈希的语法跟数组是一样的, 像下面这样写:

```
match => [ "field1", "pattern1", "field2", "pattern2" ]
```

#4.字段引用(field reference)

字段是 `Logstash::Event` 对象的属性。我们之前提过事件就像一个哈希一样, 所以你可以想象字段就像一个键值对。

小贴士: 我们叫它字段, 因为 *Elasticsearch* 里是这么叫的。

如果你想在 Logstash 配置中使用字段的值，只需要把字段的名称写在中括号 [] 里就行了，这就叫字段引用。

对于**嵌套字段**(也就是多维哈希表，或者叫哈希的哈希)，每层的字段名都写在 [] 里就可以了。比如，你可以从 geoip 里这样获取 longitude 值(是的，这是个笨办法，实际上有单独的字段专门存这个数据的)：

```
[geoip][location][0]
```

小贴士：logstash 的数组也支持倒序下标，即 `[geoip][location][-1]` 可以获取数组最后一个元素的值。

Logstash 还支持变量内插，在字符串里使用字段引用的方法是这样：

```
"the longitude is %{{geoip}[location][0]}"
```

#5. 条件判断(condition)

Logstash 从 1.3.0 版开始支持条件判断和表达式。

表达式支持下面这些操作符：

- ==(等于), !=(不等于), <(小于), >(大于), <=(小于等于), >=(大于等于)
- =~(匹配正则), !~(不匹配正则)
- in(包含), not in(不包含)
- and(与), or(或), nand(非与), xor(非或)
- ()(复合表达式), !()(对复合表达式结果取反)

通常来说，你都会在表达式里用到字段引用。为了尽量展示全面各种表达式，下面虚拟一个示例：

```
if "_grokparsefailure" not in [tags] {  
} else if [status] !~ /^2\d\d/ or ( [url] == "/noc.gif" nand [geoip][city] != "beijing" ) {  
} else {  
}
```

#6. 命令行参数

Logstash 提供了一个 shell 脚本叫 logstash 方便快速运行。它支持以下参数：

- -e

意即执行。我们在 "Hello World" 的时候已经用过这个参数了。事实上你可以不写任何具体配置，直接运行 `bin/logstash -e "` 达到相同效果。这个参数的默认值是下面这样：

```
input {  
  stdin { }  
}  
output {  
  stdout { }  
}
```

- --config 或 -f

意即文件。真实运用中，我们会写很长的配置，甚至可能超过 shell 所能支持的 1024 个字符长度。所以我们必把配置固化到文件里，然后通过 `bin/logstash -f agent.conf` 这样的形式来运行。

此外，logstash 还提供一个方便我们规划和书写配置的小功能。你可以直接用 `bin/logstash -f /etc/logstash.d/` 来运行。logstash 会自动读取 `/etc/logstash.d/` 目录下所有 *.conf 的文本文件，然后在自己内存里拼接成一个完整的大配置文件，再去执行。

注意:

logstash 列出目录下所有文件时,是字母排序的。而 logstash 配置段的 filter 和 output 都是顺序执行,所以顺序非常重要。采用多文件管理的用户,推荐采用数字编号方式命名配置文件,同时在配置中,严谨采用 if 判断限定不同日志的动作。

- --configtest 或 -t

意即测试。用来测试 Logstash 读取到的配置文件语法是否能正常解析。Logstash 配置语法是用 grammar.treetop 定义的。尤其是使用了上一条提到的读取目录方式的读者,尤其要提前测试。

- --log 或 -l

意即日志。Logstash 默认输出日志到标准错误。生产环境下你可以通过 bin/logstash -l logs/logstash.log 命令来统一存储日志。

- --pipeline-workers 或 -w

运行 filter 和 output 的 pipeline 线程数量。默认是 CPU 核数。

- --pipeline-batch-size 或 -b

每个 Logstash pipeline 线程,在执行具体的 filter 和 output 函数之前,最多能累积的日志条数。默认是 125 条。越大性能越好,同样也会消耗越多的 JVM 内存。

- --pipeline-batch-delay 或 -u

每个 Logstash pipeline 线程,在打包批量日志的时候,最多等待几毫秒。默认是 5 ms。

- --pluginpath 或 -P

可以写自己的插件,然后用 bin/logstash --pluginpath /path/to/own/plugins 加载它们。

- --verbose

输出一定的调试日志。+

- --debug

输出更多的调试日志。

#7.设置文件

从 Logstash 5.0 开始,新增了 \$LS_HOME/config/logstash.yml 文件,可以将所有的命令行参数都通过 YAML 文件方式设置。同时为了反映命令行配置参数的层级关系,参数也都改成用.而不是-了。

```
pipeline:
```

```
  workers: 24
```

```
  batch:
```

```
    size: 125
```

```
    delay: 5
```

3.input 配置

input 及输入是指日志数据传输到 Logstash 中。其中常见的配置如下:

- file: 从文件系统中读取一个文件,很像 UNIX 命令 "tail -0a"

- stdin: logstash 里最简单和基础的插件了,一个输入界面让你手工输入东西

- syslog: 监听 514 端口,按照 RFC3164 标准解析日志数据

- redis: 从 redis 服务器读取数据,支持 channel(发布订阅)和 list 模式。redis 一般在 Logstash 消费集群中作为"broker"角色,保存 events 队列供 Logstash 消费。

- lumberjack: 使用 lumberjack 协议来接收数据,目前已经改为 logstash-forwarder。

具体的 Input 插件详见官网: <https://www.elastic.co/guide/en/logstash/current/input-plugins.html>

下面取几个简单的例子讲一下

#1.file 文件例子

第一章第 5 小点中的“#3.配置 pipeline 文件”已经有，退回看一下即可。

#2.Stdin 例子

```
cat>>stdin.conf<<EOF
input {
  stdin {
    add_field => {"key" => "value"}
    codec => "plain"
    tags => ["add"]
    type => "std"
  }
}
output{
  stdout {
    codec=>rubydebug
  }
}
EOF
cat stdin.conf
logstash -f t1.conf
#然后过一会再输入 hello world
```

```
[root@vm1 conf.d]# logstash -f stdin.conf
Sending Logstash's logs to /disk1/logs/logstash which is now configured via log4j2.properties
The stdin plugin is now waiting for input:
hello world
{
  "@timestamp" => 2017-09-18T09:54:55.683Z,
  "@version" => "1",
  "host" => "0.0.0.0",
  "message" => "hello world",
  "type" => "std",
  "key" => "value",
  "tags" => [
    [0] "add"
  ]
}
```

数据在线程之间以 事件 的形式流传。不要叫行，因为 logstash 可以处理多行事件。

Logstash 会给事件添加一些额外信息。最重要的就是 **@timestamp**，用来标记事件的发生时间。因为这个字段涉及到 Logstash 的内部流转，所以必须是一个 joda 对象，如果你尝试自己给一个字符串字段重命名为 **@timestamp** 的话，Logstash 会直接报错。所以，请使用 [filters/date 插件](#) 来管理这个特殊字段。

此外，大多数时候，还可以见到另外几个：

1. **host** 标记事件发生在哪里。
2. **type** 标记事件的唯一类型。

3. tags 标记事件的某方面属性。这是一个数组，一个事件可以有多个标签。

你可以随意给事件添加字段或者从事件里删除字段。事实上事件就是一个 **Ruby** 对象，或者更简单的理解为就是一个哈希也行。

*小贴士：每个 **logstash** 过滤插件，都会有四个方法叫 **add_tag**, **remove_tag**, **add_field** 和 **remove_field**。它们在插件过滤匹配成功时生效。*

#3.rsyslog

#1) 客户端配置

#随便修改为接受远程连接，这里是 ELK 服务器 192.168.3.11

```
sed -i '/authpriv.none/s/^/#/' /etc/rsyslog.conf
sed -i '/^authpriv/s/^/#/' /etc/rsyslog.conf
sed -i '/remote-host/s/#/' /etc/rsyslog.conf
egrep 'authpriv.none;cron.none|^authpriv|remote-hos' /etc/rsyslog.conf
sed -i 's/remote-host/192.168.3.11/' /etc/rsyslog.conf
grep '192.168.3.11' /etc/rsyslog.conf
#重启
/etc/init.d/rsyslog restart
```

#2) 编写配置 logstash 的 rsyslog 配置

#在 ELK 服务器上做如下操作

```
systemctl stop rsyslog
systemctl status rsyslog
#为了方便查看结果我把调试打开
cat>>rsyslog.conf<<EOF
input {
  tcp {
    port => 514
    type => syslog
  }
  udp {
    port => 514
    type => syslog
  }
}
output {
  elasticsearch {
    #hosts => ["192.168.3.11:9200","192.168.3.12:9200"]
    hosts => ["192.168.3.11:9200"]
    user => "elastic"
    password => "123456"
    index => "messages-%{+YYYY.MM.dd}"
  }
  stdout {
```



```

    codec => rubydebug
  }
}
cat rsyslog.conf
logstash -f rsyslog.conf

```

#从配置文件上分析 input 中的 type 为 ES 中的 type，output 中的索引名 index 为 messages+当前日期

#3) 在客户端用命令测试一下结果

```
logger -p info "hello, remote rsyslog"
```

在 ELK 的界面看到了如下字样，说明成功了

```

{
  "@version" => "1",
  "host" => "192.168.3.75",
  "@timestamp" => 2017-09-18T10:49:22.510Z,
  "message" => "<14>Sep 18 18:49:22 vm5 root: hello, remote rsyslog",
  "type" => "syslog",
  "port" => 33199
}

```

也可以登陆 kibana 的开发工具查看

```

1 GET _search
2 {
3   "query": {
4     "match_phrase_prefix": {
5       "message": "hello, remote rsyslog"
6     }
7   }
8 }
9
10 {
11   "took": 27,
12   "timed_out": false,
13   "_shards": {
14     "total": 71,
15     "successful": 71,
16     "skipped": 0,
17     "failed": 0
18   },
19   "hits": {
20     "total": 1,
21     "max_score": 0.72772545,
22     "hits": [
23       {
24         "_index": "messages-2017.09.18",
25         "_type": "syslog",
26         "_id": "AV6UmlmpcPj0o-nrX_3",
27         "_score": 0.72772545,
28         "_source": {
29           "@version": "1",
30           "host": "192.168.3.75",
31           "@timestamp": "2017-09-18T10:49:22.510Z",
32           "message": "<14>Sep 18 18:49:22 vm5 root: hello, remote rsyslog",
33           "type": "syslog",
34           "port": 33199
35         }
36       }
37     ]
38   }
39 }

```

从上面的结果可以看出，索引和类型与 logstash 配置中的 “messages-%{+YYYY.MM.dd}” 和 “type” 是一致的。

4.codec 配置

Codec 是 logstash 从 1.3.0 版开始新引入的概念(Coec 来自 Coder/decoder 两个单词的首字母缩写)。

在此之前，logstash 只支持纯文本形式输入，然后以过滤器处理它。但现在，我们可以在输入期处理不同类型的数据，这全是因为有了 codec 设置。

所以，这里需要纠正之前的一个概念。Logstash 不只是一个 input | filter | output 的数据流，而是一个 input | decode | filter | encode | output 的数据流！codec 就是用来

decode、encode 事件的。

codec 的引入，使得 logstash 可以更好更方便的与其他有自定义数据格式的运维产品共存，比如 graphite、fluent、netflow、collectd，以及使用 msgpack、json、edn 等通用数据格式的其他产品等。

常用的 codec 配置如下：

- **json**：使用 json 格式对数据进行编码/解码
- **multiline**：将汇多个事件中数据汇总为一个单一的行。比如：java 异常信息和堆栈信息
- **collectd**：是一个守护(daemon)进程，用来收集系统性能和提供各种存储方式来存储不同值的机制。它会在系统运行和存储信息时周期性的统计系统的相关统计信息。利用这些信息有助于查找当前系统性能瓶颈（如作为性能分析 performance analysis）和预测系统未来的 load（如能力部署 capacity planning）等
- **netflow**：用 netflow 格式对数据进行编码/解码，主要用网络流量方面。
- **rubydebug**：当输出方式为 RubyDebug 情况下，Logstash 会自动输出一个 JSON 格式的结果。并自动添加了当前的系统时间，版本号，及 Host 信息。

具体的 codec 插件详见官网：<https://www.elastic.co/guide/en/logstash/current/codec-plugins.html>

#1.json 数据例子

我们可以把 nginx 日志格式配置成 json 格式，然后通过 logstash 分析，当然最后也可以输入到 ES 中，这里我不当输入。操作如下：

#1) 配置 nginx 日志格式为 json

为了方便我在 ELK 服务器上直接 yum 安装了 nginx

```
#rpm -ivh http://nginx.org/packages/centos/6/noarch/RPMS/nginx-release-centos-6-0.el6ngx.noarch.rpm
rpm -ivh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7ngx.noarch.rpm
yum install nginx -y
cd /etc/nginx/
vim nginx.conf
#在 log_format 处插入 json 格式
```

```
log_format logstash_json '{ "@timestamp": "$time_local", '
    "@fields": { '
        "remote_addr": "$remote_addr", '
        "remote_user": "$remote_user", '
        "body_bytes_sent": "$body_bytes_sent", '
        "request_time": "$request_time", '
        "status": "$status", '
        "request": "$request", '
        "request_method": "$request_method", '
        "http_referrer": "$http_referer", '
        "body_bytes_sent": "$body_bytes_sent", '
        "http_x_forwarded_for": "$http_x_forwarded_for", '
        "http_user_agent": "$http_user_agent" } }';
```

```

http {
    include      /etc/nginx/mime.types;
    default_type application/octet-stream;

    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

    log_format logstash_json '{ "@timestamp": "$time_local", '
                              '"@fields": { '
                                '"remote_addr": "$remote_addr", '
                                '"remote_user": "$remote_user", '
                                '"body_bytes_sent": "$body_bytes_sent", '
                                '"request_time": "$request_time", '
                                '"status": "$status", '
                                '"request": "$request", '
                                '"request_method": "$request_method", '
                                '"http_referrer": "$http_referer", '
                                '"body_bytes_sent": "$body_bytes_sent", '
                                '"http_x_forwarded_for": "$http_x_forwarded_for", '
                                '"http_user_agent": "$http_user_agent" } }';

    access_log /var/log/nginx/access.log main;

```

#然后把默认站点的日志格式修改为 logstash_json 就行了

```

cd conf.d/
vim default.conf
access_log /disk1/logs/nginx/localhost.access.log logstash_json;
#建立日志目录
mkdir -p /disk1/logs/nginx
chown nginx.nginx -R /disk1/logs/nginx
#重启 nginx
nginx -t
systemctl start nginx
systemctl status nginx

```

#上面配置完之后，访问一下 nginx，再查看一下日志是否生成 json 格式

#2) 配置 logstash 配置的 nginx 配置文件

```

#t1 目录是我自己建立的
cd /etc/logstash/t1/
cat>>nginx.conf<<EOF
input {
    file {
        path => "/disk1/logs/nginx/localhost.access.log"
        codec => "json"
    }
}
output{
    stdout {

```

```

        codec=>rubydebug
    }
}
EOF
cat nginx.conf
logstash -f nginx.conf

```

#测试

打开浏览器输入 nginx 地址 http://192.168.3.11，然后回到 ELK 界面查看变化。

```

[root@vm1 t1]# logstash -f nginx.conf
Sending Logstash's logs to /disk1/logs/logstash which is now configured via log4j2.properties
{
  "path" => "/disk1/logs/nginx/localhost.access.log",
  "@timestamp" => 2017-09-19T07:33:38.922Z,
  "_@timestamp" => "19/Sep/2017:15:33:38 +0800",
  "@version" => "1",
  "host" => "0.0.0.0",
  "@fields" => {
    "remote_user" => "-",
    "remote_addr" => "192.168.3.200",
    "request" => "GET / HTTP/1.1",
    "request_time" => "0.000",
    "body_bytes_sent" => "612",
    "http_x_forwarded_for" => "-",
    "http_referer" => "-",
    "request_method" => "GET",
    "status" => "200",
    "http_user_agent" => "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:55.0) Gecko/20100101 Firefox/55.0"
  },
  "tags" => [
    [0] "_timestampparsefailure"
  ]
}

```

#2. Multiline 合并多行数据例子

```

input {
  stdin {
    codec =>multiline {
      charset=>...      #可选      字符编码
      max_bytes=>...     #可选      bytes 类型      设置最大的字节数
      max_lines=>...     #可选      number 类型     设置最大的行数,默认是 500 行
      multiline_tag...   #可选      string 类型     设置一个事件标签,默认是 multiline
      pattern=>...       #必选      string 类型     设置匹配的正则表达式
      patterns_dir=>...  #可选      array 类型      可以设置多个正则表达式
      negate=>...        #可选      boolean 类型    设置 true 是向前匹配
                                          设置 false 向后匹配,默认是 FALSE
      what=>...          #必选      置未匹配的内容是向前合并还是先后合并,previous,next 两个值选择
    }
  }
}

```

```
cat>>multiline.conf<<EOF
```

```

input {
  stdin {
    codec => multiline {
      pattern => "^\\["
      negate => true
      what => "previous"
    }
  }
}
output{
  stdout {
    codec=>rubydebug
  }
}
EOF
cat multiline.conf
logstash -f multiline.conf

```

#等下输入几行数据

```

[Aug/08/08 14:54:03] hello world
[Aug/08/09 14:54:04] hello logstash
    hello best practice
    hello raochenlin
[Aug/08/10 14:54:05] the end

```

#运行结果如下:

```

[root@vml t1]# logstash -f multiline.conf
Sending Logstash's logs to /disk1/logs/logstash which is now configured via log4j2.properties
The stdin plugin is now waiting for input:

[Aug/08/08 14:54:03] hello world
[Aug/08/09 14:54:04] hello logstash
    hello best practice
    hello raochenlin
[Aug/08/10 14:54:05] the end
{
  "@version" => "1",
  "host" => "0.0.0.0",
  "@timestamp" => 2017-09-21T16:33:03.722Z,
  "message" => "[Aug/08/08 14:54:03] hello world"
}
{
  "@version" => "1",
  "host" => "0.0.0.0",
  "@timestamp" => 2017-09-21T16:33:03.770Z,
  "message" => "[Aug/08/09 14:54:04] hello logstash\n    hello best practice\n    hello raochenlin",
  "tags" => [
    [0] "multiline"
  ]
}

```

5.filter 配置

Filters 在 Logstash 处理链中担任中间处理组件。他们经常被组合起来实现一些特定的行为来，处理匹配特定规则的事件流。常见的 filters 如下：

- **grok**: 解析无规则的文字并转化为有结构的格式。**Grok** 是目前最好的方式来将无结构的数据转换为有结构可查询的数据。有 120 多种匹配规则，会有一种满足你的需要。
- **mutate**: **mutate filter** 允许改变输入的文档，你可以从命名，删除，移动或者修改字段在处理事件的过程中。
- **drop**: 丢弃一部分 **events** 不进行处理，例如：**debug events**。
- **clone**: 拷贝 **event**，这个过程中也可以添加或移除字段。
- **geoip**: 添加地理信息(为前台 **kibana** 图形化展示使用)

具体的 **filter** 插件详见官网：<https://www.elastic.co/guide/en/logstash/current/filter-plugins.html>

#1.date

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

时间处理

之前章节已经提过，**filters/date** 插件可以用来转换你的日志记录中的时间字符串，变成 **LogStash::Timestamp** 对象，然后转存到 **@timestamp** 字段里。

注意：因为在稍后的 **outputs/elasticsearch** 中常用的 **%{+YYYY.MM.dd}** 这种写法必须读取 **@timestamp** 数据，所以一定不要直接删掉这个字段保留自己的字段，而是应该用 **filters/date** 转换后删除自己的字段！

这在导入旧数据的时候固然非常有用，而在实时数据处理的时候同样有效，因为一般情况下数据流程中我们都会有缓冲区，导致最终的实际处理时间跟事件产生时间略有偏差。

小贴士：个人强烈建议打开 **Nginx** 的 **access_log** 配置项的 **buffer** 参数，对极限响应性能有极大提升！

配置示例

filters/date 插件支持五种时间格式：

- **ISO8601**

类似 "2011-04-19T03:44:01.103Z" 这样的格式。具体 **Z** 后面可以有 "08:00" 也可以没有，".103" 这个也可以没有。常用场景里来说，**Nginx** 的 **log_format** 配置里就可以使用 **\$time_iso8601** 变量来记录请求时间成这种格式。

- **UNIX**

UNIX 时间戳格式，记录的是从 1970 年起始至今的总秒数。**Squid** 的默认日志格式中就使用了这种格式。

- **UNIX_MS**

这个时间戳则是从 1970 年起始至今的总毫秒数。据我所知，**JavaScript** 里经常使用这个时间格式。

- **TAI64N**

TAI64N 格式比较少见，是这个样子的：**@4000000052f88ea32489532c**。我目前只知道常见应用中，**qmail** 会用这个格式。

- **Joda-Time** 库

Logstash 内部使用了 **Java** 的 **Joda** 时间库来作时间处理。所以我们可以使用 **Joda** 库所支持的时间格式来作具体定义。**Joda** 时间格式定义见下表：

详见链接：<https://kibana.logstash.es/content/logstash/plugins/filter/date.html>

时区问题的解释

很多中国用户经常提一个问题：为什么 **@timestamp** 比我们晚了 8 个小时？怎么修改成北京时间？

其实，Elasticsearch 内部，对时间类型字段，是**统一采用 UTC 时间，存成 long 长整形数据的**！对日志统一采用 UTC 时间存储，是国际安全/运维界的一个通识——欧美公司的服务器普遍广泛分布在多个时区里——不像中国，地域横跨五个时区却只用北京时间。

对于页面查看，ELK 的解决方案是在 Kibana 上，读取浏览器的当前时区，然后在页面上转换时间内容的**显示**。

所以，建议大家接受这种设定。否则，即便你用 `.getLocalTime` 修改，也还要面临在 Kibana 上反过去修改，以及 Elasticsearch 原有的 `["now-1h" TO "now"]` 这种方便的搜索语句无法正常使用的尴尬。

以上，请读者自行斟酌。

#2.grok 正则捕获

参照地址：<https://kibana.logstash.es/content/logstash/plugins/filter/grok.html>

Grok 表达式语法

Grok 支持把预定义的 *grok* 表达式 写入到文件中，官方提供的预定义 *grok* 表达式见：

<https://github.com/logstash-plugins/logstash-patterns-core/tree/master/patterns>。

注意：在新版本的 *logstash* 里面，*pattern* 目录已经为空，最后一个 *commit* 提示 *core patterns* 将会由 *logstash-patterns-core gem* 来提供，该目录可供用户存放自定义 *patterns*

下面是从官方文件中摘抄的最简单但是足够说明用法的示例：

```
USERNAME [a-zA-Z0-9._-]+
USER %{USERNAME}
```

第一行，用普通的正则表达式来定义一个 *grok* 表达式；第二行，通过打印赋值格式(*sprintf format*)，用前面定义好的 *grok* 表达式来定义另一个 *grok* 表达式。

grok 表达式的打印赋值格式的完整语法是下面这样的：

```
%{PATTERN_NAME:capture_name:data_type}
```

小贴士：*data_type* 目前只支持两个值：*int* 和 *float*。

所以我们可以改进我们的配置成下面这样：

```
input {stdin{}}
filter {
  grok {
    match => {
      "message" => "%{WORD} %{NUMBER:request_time:float} %{WORD}"
    }
  }
}
output {stdout{codec => rubydebug}}
```

#运行 *logstash* 进程然后输入 "begin 123.456 end"

```
[root@vm1 t1]# logstash -f data.conf
Sending Logstash's logs to /disk1/logs/logstash which is now configured via log4j2.properties
The stdin plugin is now waiting for input:
begin 123.456 end
{
  "@version" => "1",
  "host" => "0.0.0.0",
  "@timestamp" => 2017-09-22T09:18:09.496Z,
  "request_time" => 123.456,
  "message" => "begin 123.456 end"
}
```

#3.dissect

grok 作为 Logstash 最广为人知的插件，在性能和资源损耗方面同样也广为诟病。为了应对这个情况，同时也考虑到大多数时候，日志格式并没有那么复杂，Logstash 开发团队在 5.0 版新添加了另一个解析字段的插件：dissect。

当日志格式有比较简明的分隔标志位，而且重复性较大的时候，我们可以使用 dissect 插件更快的完成解析工作。下面是解析 syslog 的示例：

```
input {stdin{}}
filter {
  dissect {
    mapping => {
      "message" => "%{ts} %{+ts} %{+ts} %{src} %{} %{prog}[%{pid}]: %{msg}"
    }
    convert_datatype => {
      pid => "int"
    }
  }
}
output {stdout{codec => rubydebug}}
```

语法解释：

我们看到上面使用了和 Grok 很类似的 %{} 语法来表示字段，这显然是基于习惯延续的考虑。不过示例中 %{+ts} 的加号就不一般了。dissect 除了字段外面的字符串定位功能以外，还通过几个特殊符号来处理字段提取的规则：

- %{+key} 这个 + 表示，前面已经捕获到一个 key 字段了，而这次捕获的内容，自动添补到之前 key 字段内容的后面。
- %{+key/2} 这个 /2 表示，在有多次捕获内容都填到 key 字段里的时候，拼接字符串的顺序谁前谁后。/2 表示排第 2 位。
- %{?string} 这个 ? 表示，这块只是一个占位，并不会实际生成捕获字段存到 Event 里面。
- %{?string} %{&string} 当同样捕获名称都是 string，但是一个 ? 一个 & 的时候，表示这是一个键值对。

比如对 `http://rizhiyi.com/index.do?id=123` 写这么一段配置：

```
http://%{domain}%{url}?%{?arg1}=%{&arg1}
```

则最终生成的 Event 内容是这样的：


```
{
  domain => "rizhiyi.com",
  id => "123"
}
```

#4.其它

请看链接 <https://kibana.logstash.es/content/logstash/plugins/filter/json.html>

6.output

outputs 是 logstash 处理管道的最末端组件。一个 event 可以在处理过程中经过多重输出，但是一旦所有的 outputs 都执行结束，这个 event 也就完成生命周期。一些常用的 outputs 包括：

- **elasticsearch**: 如果你计划将高效的保存数据，并且能够方便和简单的进行查询...Elasticsearch 是一个好的方式。是的，此处有做广告的嫌疑。
- **file**: 将 event 数据保存到文件中。
- **graphite**: 将 event 数据发送到图形化组件中，一个很流行的开源存储图形化展示的组件。<http://graphite.wikidot.com/>。
- **statsd**: statsd 是一个统计服务，比如技术和时间统计，通过 udp 通讯，聚合一个或者多个后台服务，如果你已经开始使用 statsd，该选项对你应该很有用。

具体的 output 插件详见官网：<https://www.elastic.co/guide/en/logstash/current/output-plugins.html>

#1.elasticsearch

Logstash 可以使用不同的协议实现完成将数据写入 Elasticsearch 的工作。在不同时期，也有不同的插件实现方式。本节以最新版为准，即主要介绍 HTTP 方式。同时也附带一些原有的 node 和 transport 方式的介绍。

例子

```
input {stdin{}}
output {
  elasticsearch {
    hosts => ["192.168.0.2:9200"]
    index => "logstash-%{type}-%{+YYYY.MM.dd}"
    document_type => "%{type}"
    flush_size => 20000
    idle_flush_time => 10
    sniffing => true
    template_overwrite => true
  }
}
output {stdout{codec => rubydebug}}
```

解释：

批量发送

在过去的版本中，主要由本插件的 `flush_size` 和 `idle_flush_time` 两个参数共同控制 Logstash 向 Elasticsearch 发送批量数据的行为。以上面示例来说：Logstash 会努力攒到 20000 条数据一次性发送出去，但是如果 10 秒钟内也没攒够 20000 条，Logstash 还是会以当前攒到的数据量发一次。

默认情况下，`flush_size` 是 500 条，`idle_flush_time` 是 1 秒。这也是很多人改大了 `flush_size` 也没能提高写入 ES 性能的原因——Logstash 还是 1 秒钟发送一次。

从 5.0 开始，这个行为有了另一个前提：`flush_size` 的大小不能超过 Logstash 运行时的命令行参数设置的 `batch_size`，否则将以 `batch_size` 为批量发送的大小。

索引名

写入的 ES 索引的名称，这里可以使用变量。为了更贴合日志场景，Logstash 提供了 `%{+YYYY.MM.dd}` 这种写法。在语法解析的时候，看到以 `+` 号开头的，就会自动认为后面是时间格式，尝试用时间格式来解析后续字符串。所以，之前处理过程中不要给自定义字段取个加号开头的名字……

此外，注意索引名中不能有大写字母，否则 ES 在日志中会报 *InvalidIndexNameException*，但是 Logstash 不会报错，这个错误比较隐晦，也容易掉进这个坑中。

轮询

Logstash 1.4.2 在 `transport` 和 `http` 协议的情况下是固定连接指定 `host` 发送数据。从 1.5.0 开始，`host` 可以设置数组，它会从节点列表选取不同的节点发送数据，达到 Round-Robin 负载均衡的效果。

小贴士

经常有同学问，为什么 Logstash 在有多个 `conf` 文件的情况下，进入 ES 的数据会重复，几个 `conf` 数据就会重复几次。其实问题原因在之前[启动参数章节](#)有提过，`output` 段顺序执行，没有对日志 `type` 进行判断的插件配置都会全部执行一次。在 `output` 段对 `type` 进行判断的语法如下所示：

```
output {
  if [type] == "nginxaccess" {
    elasticsearch { }
  }
}
```

模板

请看链接：<https://kibana.logstash.es/content/logstash/plugins/output/elasticsearch.html>
这里不多讲

#2.标准输出(Stdout)

和之前 `inputs/stdin` 插件一样，`outputs/stdout` 插件也是最基础和简单的输出插件。同样在这里简单介绍一下，作为输出插件的一个共性了解。

示例

```
output {
  stdout {
    codec => rubydebug
    workers => 2
  }
}
```

解决：

输出插件统一具有一个参数是 `workers`。Logstash 为输出做了多线程的准备。

其次是 `codec` 设置。`codec` 的作用在之前已经讲过。可能除了 `codecs/multiline`，其他 `codec` 插件本身并没有太多的设置项。所以一般省略掉后面的配置区段。换句话说。上面配置示例的完全写法应该是：

```
output {  
  stdout {  
    codec => rubydebug {  
    }  
    workers => 2  
  }  
}
```

单就 `outputs/stdout` 插件来说，其最重要和常见的用途就是调试。所以在不太有效的时候，加上命令行参数 `-vv` 运行，查看更多详细调试信息。

七、logstash 场景示例

参见链接：<https://kibana.logstash.es/content/logstash/examples/>

八、kibana

kibana 只是一个数据展示，这里我就不详解了，有兴趣的可以看一下下面链接

<https://kibana.logstash.es/content/kibana/>

附录一、参考文章

ELK 官方文档

<https://www.elastic.co/guide/index.html>

ELKstack 中文指南

<https://kibana.logstash.es/content/>

elasticsearch 5.4 中文文档

<http://cwiki.apachecn.org/display/Elasticsearch/Index+API>

flying 飞翔

QQ: 715031064

广州 linux 运维: 478477301

linux 运维菜鸟: 313184229

2017.9.22