**搭建[E](http://www.baidu.com/link?url=T0KZZbcRPEpzyNd9gDqvGg5uQObSIRSEK8xZ3ZsAA1cmeL2x5OuUIY6373_DLeTk" \t "https://www.baidu.com/_blank)LK 日志分析教程**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修改内容** | **审批人** | **日期** |
| 1.0.0 | 初始版本 | 姚广勋 | 2020.01.22 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**搭建[E](http://www.baidu.com/link?url=T0KZZbcRPEpzyNd9gDqvGg5uQObSIRSEK8xZ3ZsAA1cmeL2x5OuUIY6373_DLeTk" \t "https://www.baidu.com/_blank)LK 日志分析环境**

ELK日志分析系统,用于日志的搜集和搜索。

Elasticsearch是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。

elasticsearch-head 是用于监控 Elasticsearch 状态的客户端插件，包括数据可视化、执行增删改查操作等。安装之前确保当前系统已经安装 nodejs 即可。

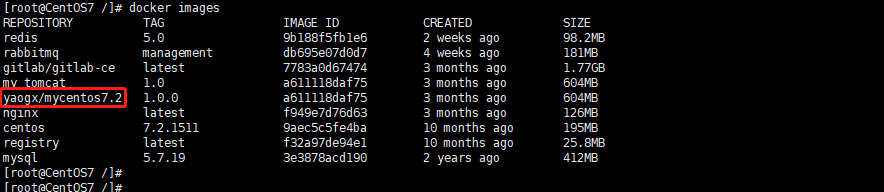
Logstash是一个完全开源的工具，他可以对你的日志进行收集、过滤，并将其存储供以后使用（如，搜索）。

Kibana 也是一个开源和免费的工具，它Kibana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。

ELK 关系：LEK：logstatsh 收集日志，存到elasticserach （存储，产生索引，搜索） 到kibana展现（view）

# 拉取CentOS7镜像

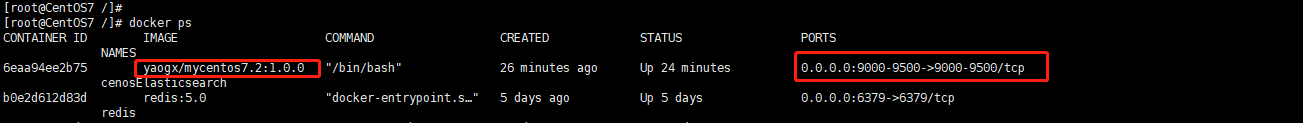
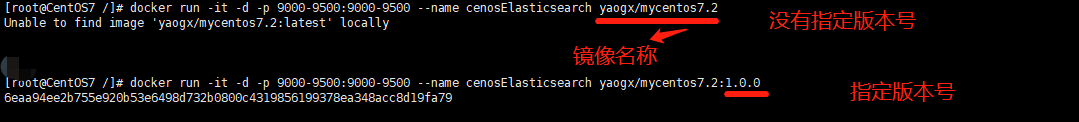
由于服务器中已经拉取，且已经配置了JDK 8 ，因此直接使用。



## 运行镜像

执行命令运行 yaogx/mycentos7.2镜像，执行命令：

docker run -it -d -p 9000-9500:9000-9500 --name cenosElasticsearch centos



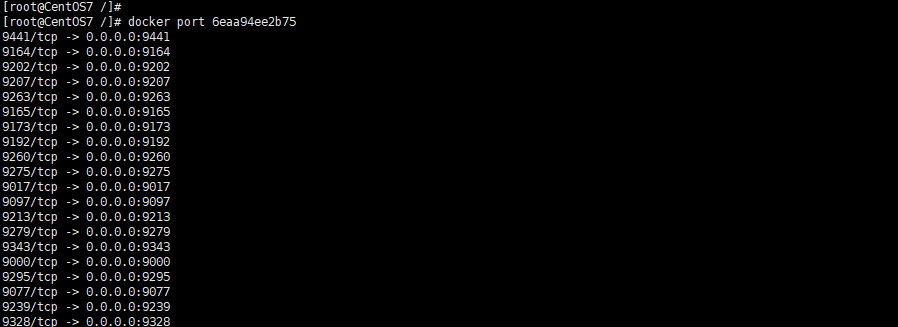
**注：**启动容器选择了**一段**ip和主机ip映射「-p 9000-9500:9000-9500」

## 查看端口号

可以使用docker port 命令查看具体映射。

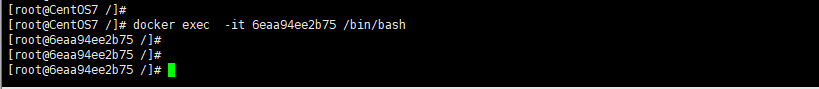
格式：docker port 容器id

执行命令：docker port 6eaa94ee2b75



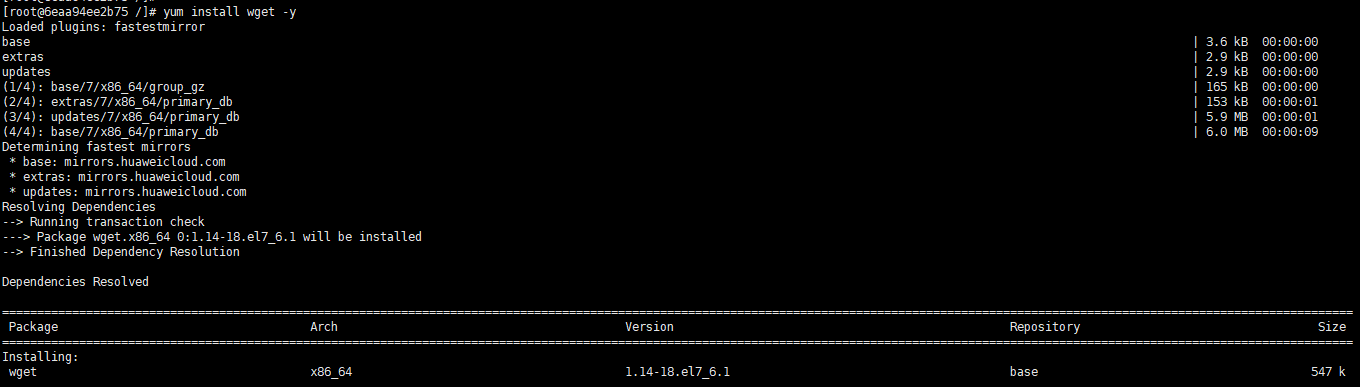
## 安装一些必要软件

**首先进入容器**，执行命令：docker exec -it 6eaa94ee2b75 /bin/bash



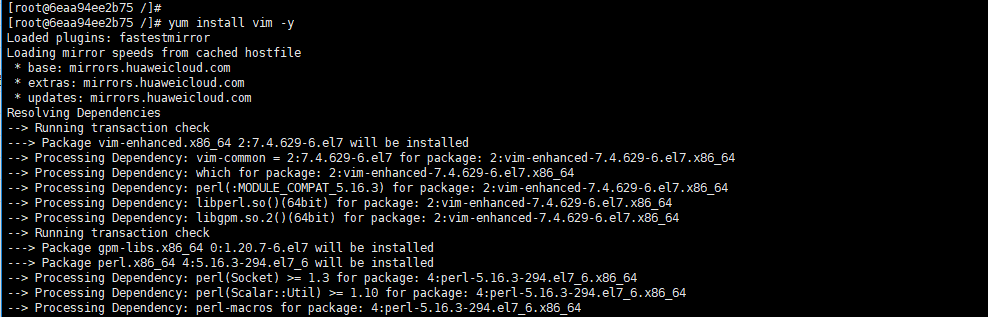
### 安装wget

执行命令：yum install wget -y



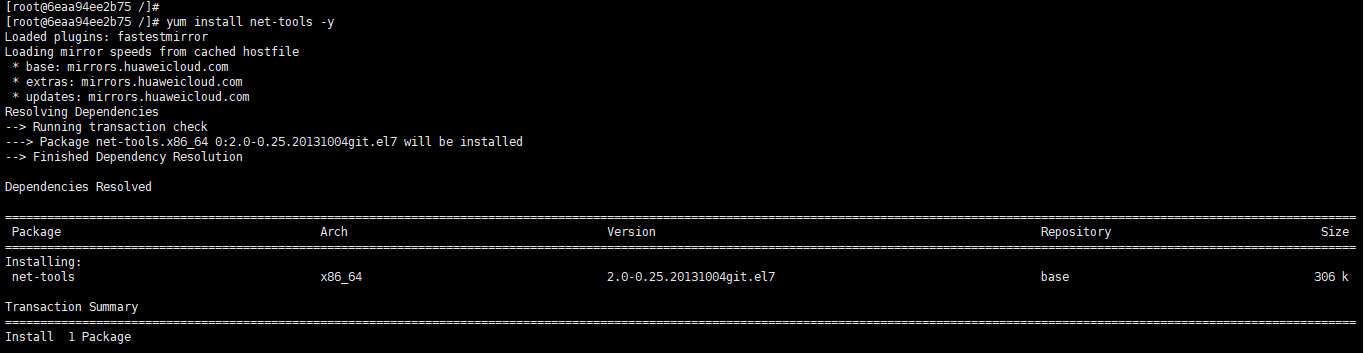
### 安装vim

执行命令：yum install vim -y



### 安装net-tools

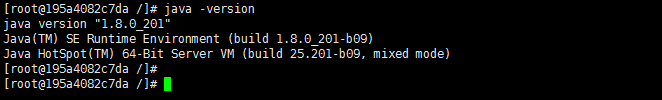
执行命令：yum install net-tools -y



# 安装**Elasticsearch**

 由于es是基于Java开发的，所以在安装es之前我得先安装jdk，要求版本在1.8以上，

在安装Elasticsearch 时，查看JDK版本，执行命令：java -version



## 创建用户

从5.0 以后，Elasticsearch 安全级别提高了，不允许使用root 账号启动，需要创建用户。

### 创建用户组及用户

创建elk 用户组，执行命令：groupadd elk



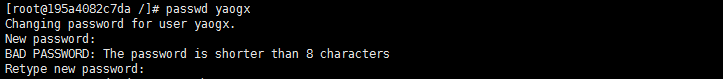
### 创建用户

执行命令：useradd yaogx



设置密码：passwd yaogx

输入密码：12345678



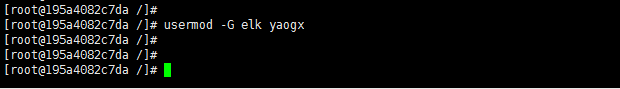
账号：yaogx

密码：12345678

### 添加用户到组

将yaogx 用户添加到elk 组

执行命令：usermod -G elk yaogx



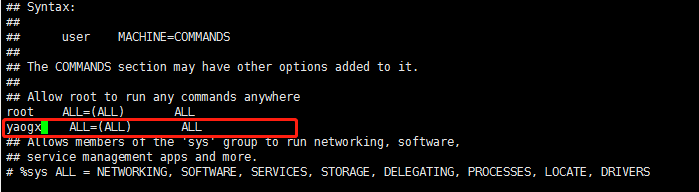
### 设置权限

设置sudo 权限，执行命令：visudo

找到root ALL=（ALL） ALL 一行，添加yaogx 用户，如下：

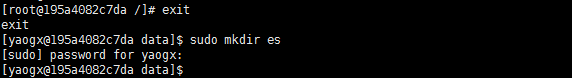
root ALL=（ALL） ALL

yaogx ALL=(ALL) ALL



然后 按 ESC 退出编辑模式，输入:wq 保存退；

输入exit,退出root用户

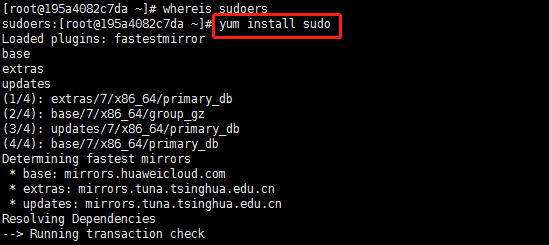


**注：如果不存在visudo**



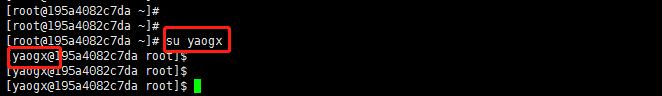
查看visudo是否存在 ，执行命令：whereis sudoers

如果不存在安装，执行命令：yum install sudo



### 切换用户

执行命令：su yaogx



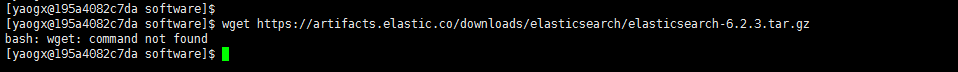
## 下载Elasticsearch

下载地址： wget <https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.2.3.tar.gz>

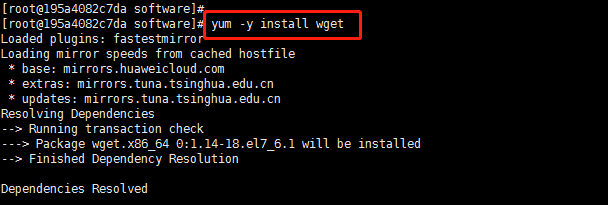
注：普通用户操作需要加：

Sudo wget <https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.2.3.tar.gz>

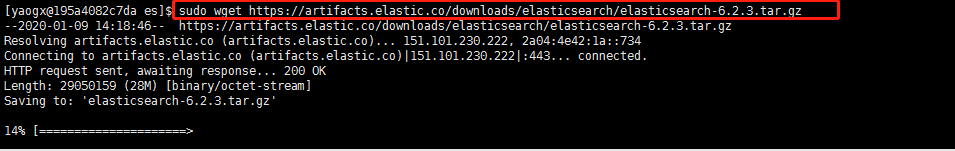
如果找提示：bash: wget: command not found



解决：执行命令：yum -y install wget



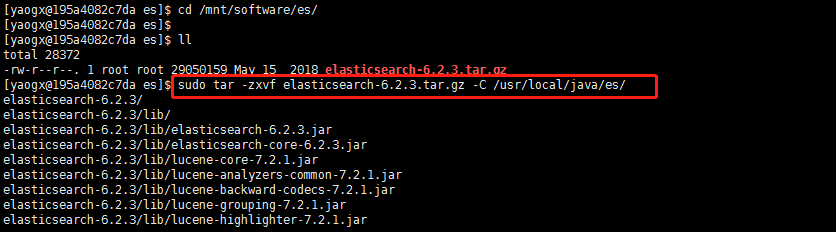
再次下载elasticsearch，如图：



### 解压elasticsearch

向指定目录解压elasticsearch。

执行命令：sudo tar -zxvf elasticsearch-6.2.3.tar.gz -C /usr/local/java/es/

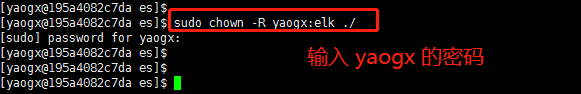


### 修改文件名

执行命令：sudo mv elasticsearch-6.2.3/ elasticsearch

### 修改目录权限

执行命令：sudo chown -R yaogx:elk ./



### 修改elasticsearch配置

1. 修改elasticsearch.yml 配置文件

执行命令：vi elasticsearch/config/elasticsearch.yml



修改：

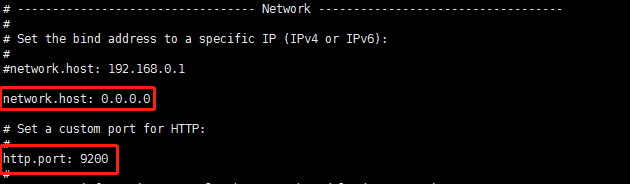
#network.host: 192.168.0.1

为：

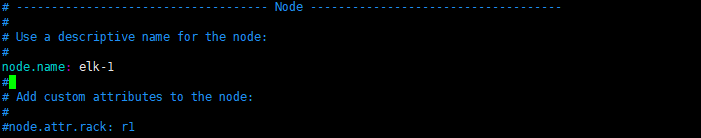
network.host: 0.0.0.0

放开端口注释：

http.port: 9200

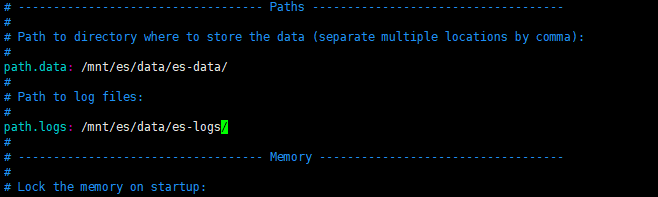
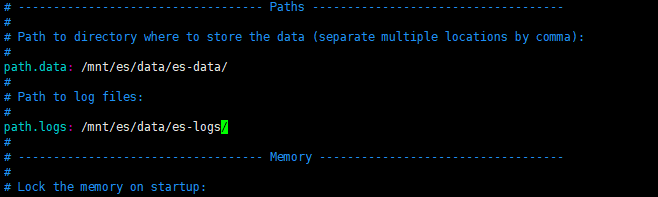


找到配置文件中的node.name，打开该配置并设置节点名称 node.name: elk-1

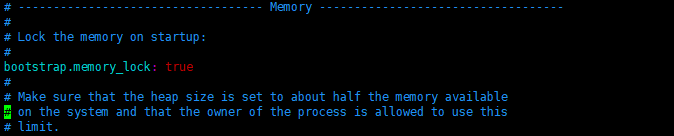


修改data存放的路径 path.data: /mnt/es/data/es-data/

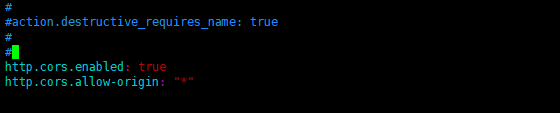
修改logs日志的路径 path.logs: /mnt/es/data/es-logs/



配置内存使用用交换分区 （暂不修改，启动报错） bootstrap.memory\_lock: true



增加新的参数，这样head插件可以访问es (5.x版本，如果没有可以自己手动加) http.cors.enabled: true http.cors.allow-origin: "\*"



注：根据需求修改

# vim /etc/elasticsearch/elasticsearch.yml 找到配置文件中的

cluster.name，打开该配置并设置集群名称

cluster.name: demon

找到配置文件中的node.name，打开该配置并设置节点名称 node.name: elk-1

修改data存放的路径 path.data: /data/es-data

修改logs日志的路径 path.logs: /var/log/elasticsearch/

配置内存使用用交换分区 bootstrap.memory\_lock: true

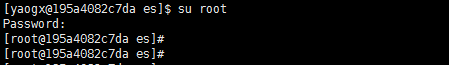
监听的网络地址 network.host: 0.0.0.0

开启监听的端口 http.port: 9200

增加新的参数，这样head插件可以访问es

(5.x版本，如果没有可以自己手动加) http.cors.enabled: true http.cors.allow-origin: "\*"

1. **切换到 root 用户**

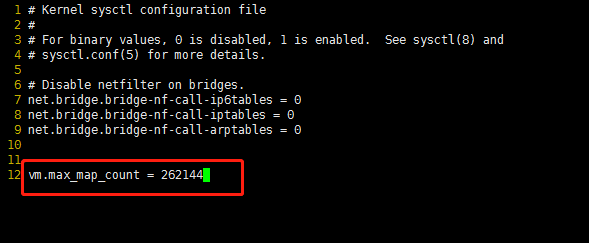


在centos7 之前，我们修改内核参数，都是使用/etc/sysctl.conf文件，但是在centos7 之后，就不再是这个文件了，如果我们想修改内核参数目前尚不清楚。

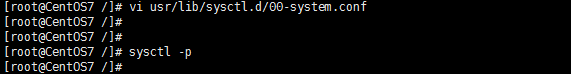
centos7 之前执行命令: vi /etc/sysctl.conf

添加：

vm.max\_map\_count = 262144



修改之后执行命令： sysctl -p

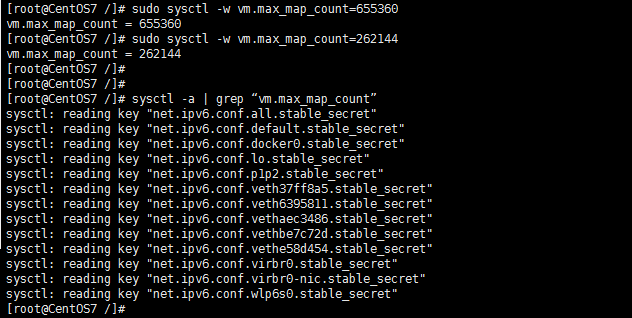


centos7 之后，这里我选择的是修改宿主机本身的配置文件，然后重新启动镜像，也能解决问题，退出容器，返回到宿主机  
修改vm.max\_map\_count 可以通过命令行修改，但是在机器重启时会失效，所以通过修改配置文件来解决问题，命令行修改办法：

sudo sysctl -w vm.max\_map\_count=262144

用以下命令查看是否修改成功：

sysctl -a | grep “vm.max\_map\_count”



修改文件: vi /etc/security/limits.conf

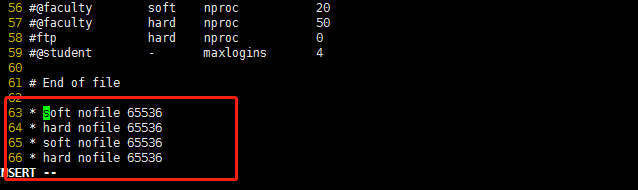
添加如下内容：

\*soft nofile 65536

\*hard nofile 65536

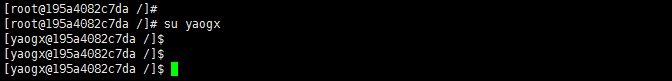
\*soft nofile 65536

\*hard nofile 65536



## 启动elasticsearch

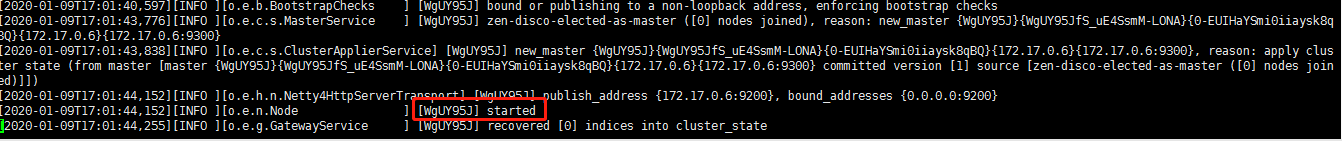
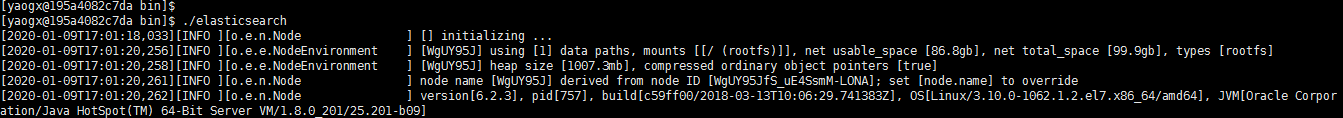
**切换用户：yaogx**



1. **启动**

**进入：cd usr/local/java/es/elasticsearch/bin/**

执行命令： ./elasticsearch

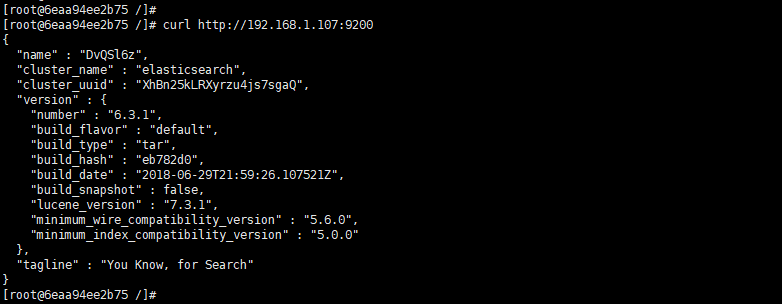


或后台启动：./elasticsearch -d

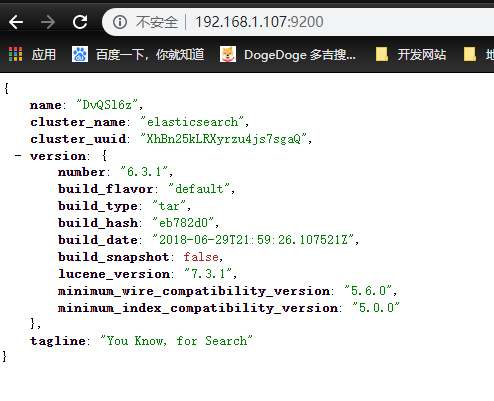


### 成功验证

执行命令：curl <http://192.168.1.107:9200>



在浏览器中输入：<http://192.168.1.107:9200/>

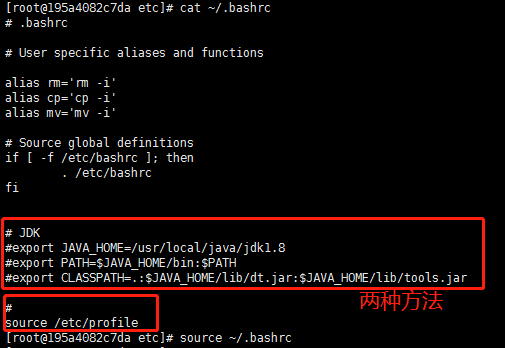


# 安装过程中的问题

## 重启容器时JDK 失效

在 ~/.bashrc文件中添加一句source /etc/profile（或者将环境变量的配置放到该文件中），就可以了

执行命令：vi ~/.bashrc

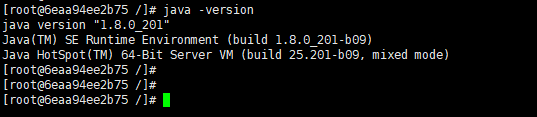


重新加载 ~/.bashrc文件 ；

执行命令：source ~/.bashrc

## 检测JDK

执行命令：java -version



# 安装elasticsearch-head

elasticsearch-head 是用于监控 Elasticsearch 状态的客户端插件，包括数据可视化、执行增删改查操作等。安装之前确保当前系统已经安装 nodejs 即可。

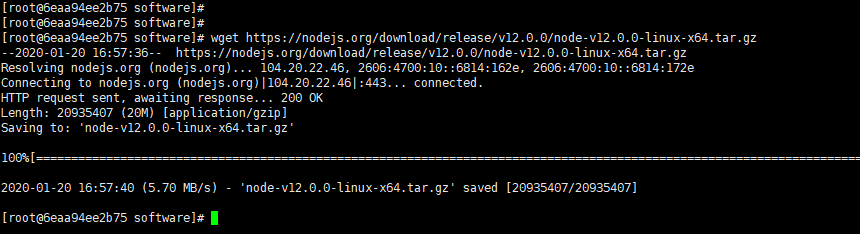
## 安装Nodejs

把当前账户切换到 ROOT 账户下，然后下载nodejs。

### 下载Nodejs

执行命令：

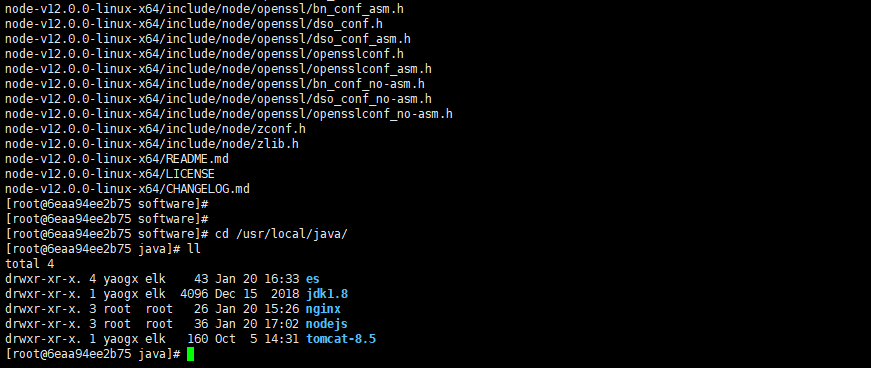
wget https://nodejs.org/download/release/v12.0.0/node-v12.0.0-linux-x64.tar.gz



### 解压nodejs

执行命令：

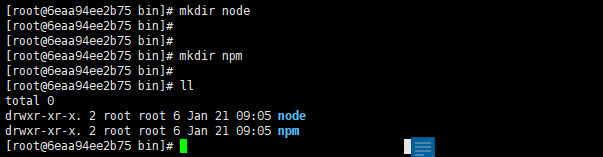
tar -zxvf node-v12.0.0-linux-x64.tar.gz -C /usr/local/java/nodejs/



### 建立软连接

由于上边安装的 Nodejs 并不完美。首先，Nodejs 的命令 node 和 npm 并不能在全局使用。通过建立软链接的方式设置为全局，为了解决这个问题，我们需要建立两个软链接。

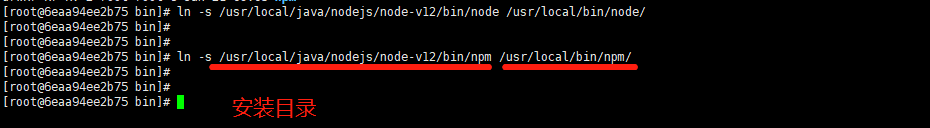
在cd /usr/local/bin/ 目录中新建 node 和npm 目录。



创建软连接，执行命令：

ln -s /usr/local/java/nodejs/node-v12/bin/node /usr/local/bin/node/

ln -s /usr/local/nodejs/bin/npm /usr/local/bin/npm/



### 配置profile 文件

执行命令：vi /etc/profile

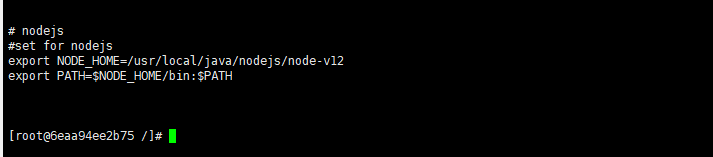
在文件末尾添加：

# nodejs

#set for nodejs

export NODE\_HOME=/usr/local/java/nodejs/node-v12

export PATH=$NODE\_HOME/bin:$PATH



重新加载 profile

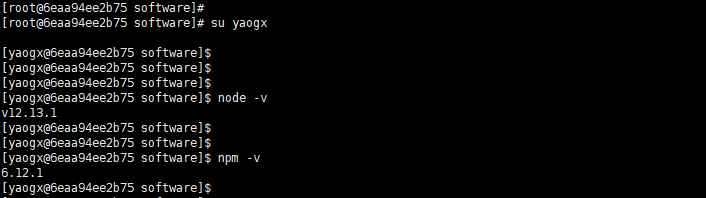
执行命令：source /etc/profile

### 验证是否安装配置成功

切换到 普通用户：su yaogx

执行命令： node -v

npm -v

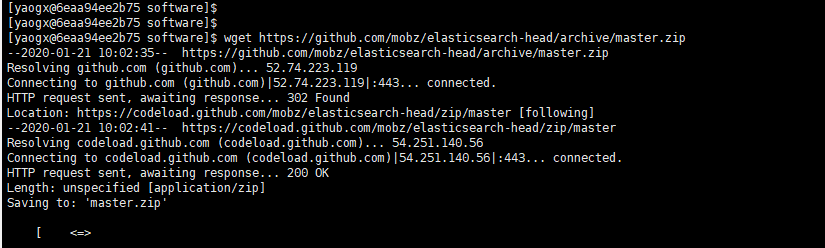


参考<https://www.jianshu.com/p/2cc020b170c0>

## 下载elasticsearch-head

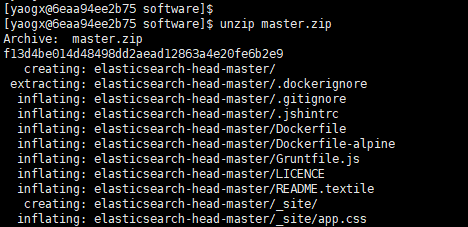
执行命令：

wget https://github.com/mobz/elasticsearch-head/archive/master.zip

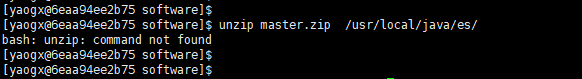


### 解压zip文件

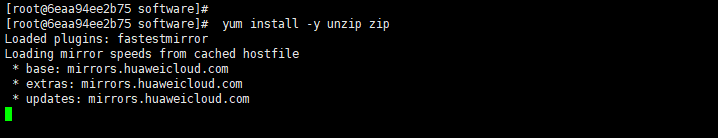
执行命令：unzip master.zip



如果出现如下情况，需要安装unzip。

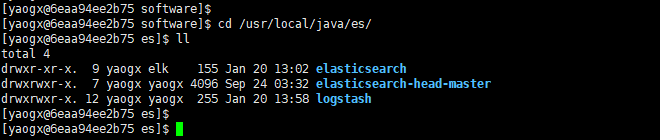
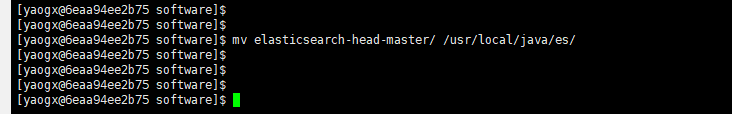


安装unzip,执行命令： yum install -y unzip zip



把解压的文件转移到指定位置 /usr/local/java/es/

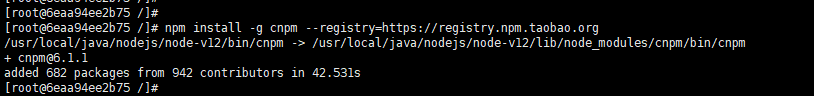
执行命令： mv elasticsearch-head-master/ /usr/local/java/es/



### 更换npm源

执行命令：

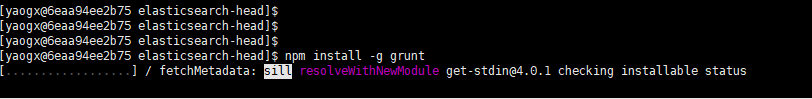
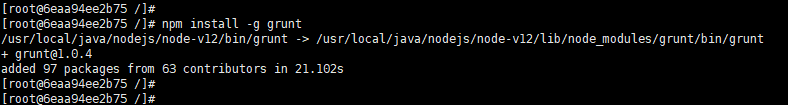
npm install -g cnpm --registry=[https://registry.npm.taobao.org](https://registry.npm.taobao.org/" \t "https://www.cnblogs.com/mkxfs/p/_blank)



### 安装grunt

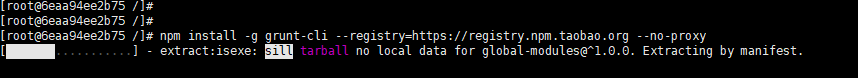
grunt是基于Node.js的项目构建工具，可以进行打包压缩、测试、执行等等工作，elasticsearch-head插件就是通过grunt启动的。

执行命令：npm install -g grunt



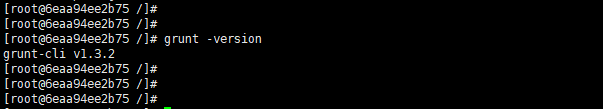
执行命令：

npm install -g grunt-cli --registry=[https://registry.npm.taobao.org](https://registry.npm.taobao.org/" \t "https://www.cnblogs.com/mkxfs/p/_blank) --no-proxy



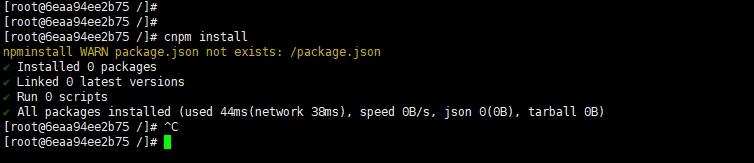
查看grunt的是否安装成功

执行命令：grunt -version



### 安装cnpm

执行命令：cnpm install



### 修改配置文件

修改elasticsearch-head 中的Gruntfile.js 文件。

找到：

|  |
| --- |
| connect: {  server: {  options: {  port: 9100,  base: '.',  keepalive: true  }  }  } |

在 添加 hostname 一项，如下：

connect: {

server: {

options: {

hostname :’0.0.0.0’,

port: 9100,

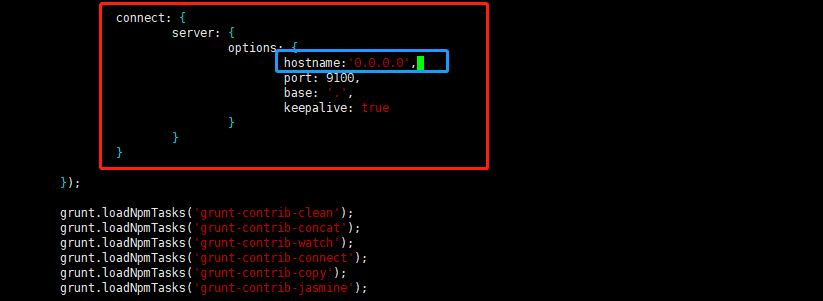
base: '.',

keepalive: true

}

}

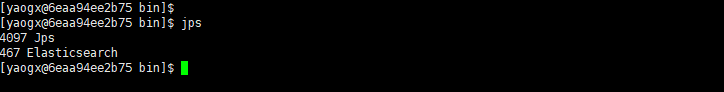
}



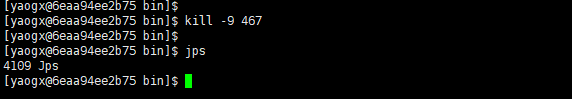
### 重启 elasticsearch

进入elasticsearch 的bin/ 目录

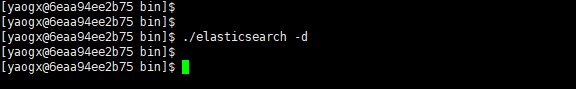
查看进程号，执行命令：jps



杀掉进程，执行命令：kill -9 467

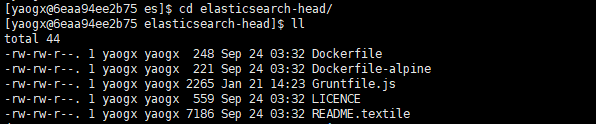


重启elasticsearch ，执行命令：./elasticsearch -d



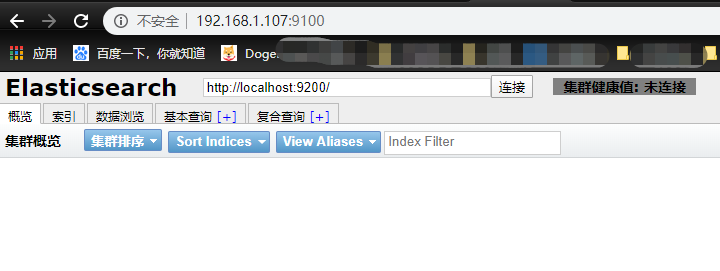
### 启动elasticsearch-head

进入 elasticsearch-head 目录下，执行启动命令。

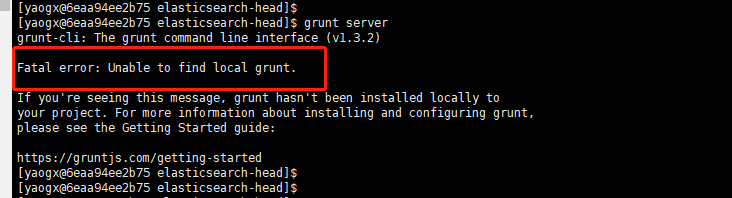


执行命令：grunt server

或 npm run start



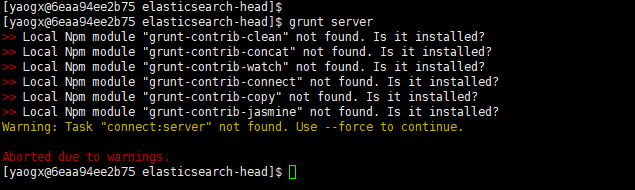
如果启动报错，如下：



解决方式：

执行命令：npm install grunt

下载后，运行grunt server会继续报错，按照报错信息进行安装：



执行命令：

npm install grunt-contrib-clean --registry=

https://registry.npm.taobao.org  
  
npm install grunt-contrib-concat --registry=

https://registry.npm.taobao.org  
  
npm install grunt-contrib-watch --registry=

<https://registry.npm.taobao.org>

npm install grunt-contrib-connect --registry=

<https://registry.npm.taobao.org>

npm install grunt-contrib-copy --registry=

<https://registry.npm.taobao.org>

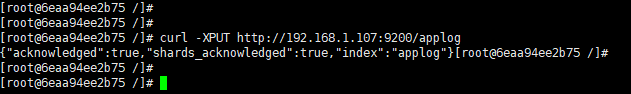
npm install grunt-contrib-jasmine --registry=

<https://registry.npm.taobao.org>

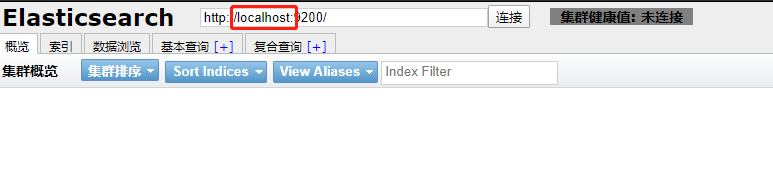
### 创建索引

在root目录下执行，

执行命令：curl -XPUT <http://192.168.1.107:9200/applog>



修改 localhost 为自己的ip



点击连接，如下图所示：



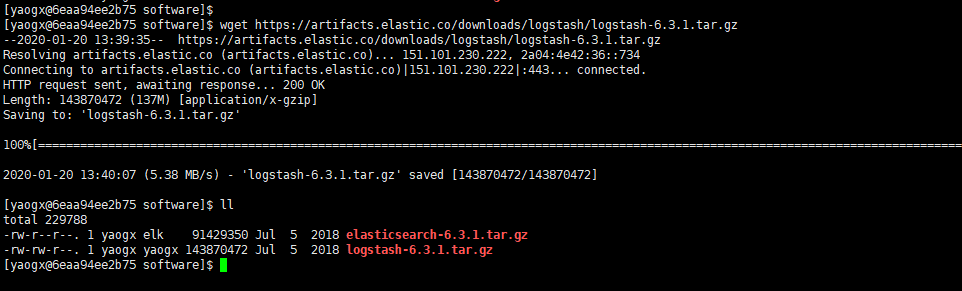
# 安装logstash

Logstash是一个完全开源的工具，他可以对你的日志进行收集、过滤，并将其存储供以后使用（如，搜索）。

## 下载logstash

执行命令：

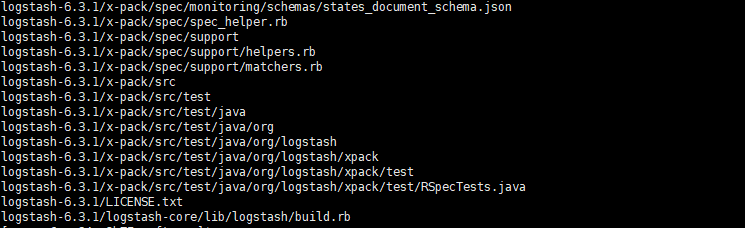
wget https://artifacts.elastic.co/downloads/logstash/logstash-6.3.1.tar.gz



### 解压logstash

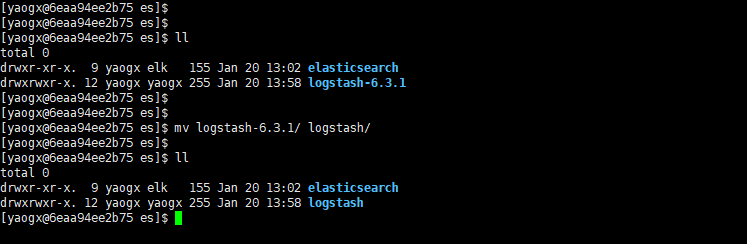
解压到指定文件夹： /usr/local/java/es/

执行命令：tar -zxvf logstash-6.3.1.tar.gz -C /usr/local/java/es/



### 修改文件名

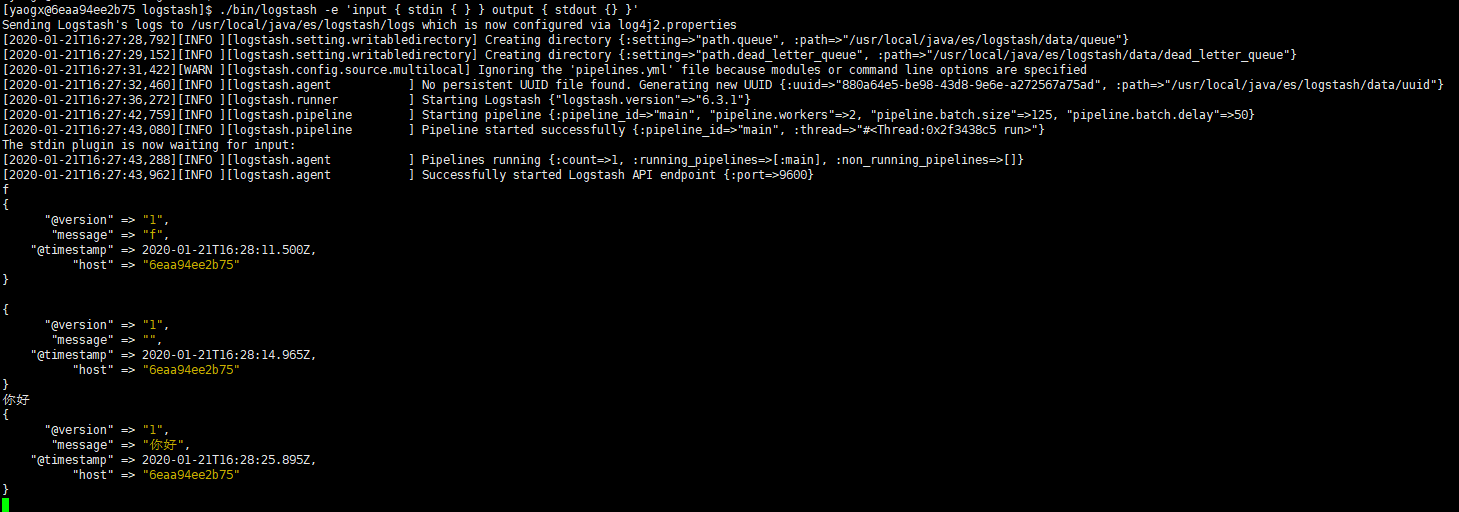
执行命令：mv logstash-6.3.1/ logstash/



### 测试

进入 cd /usr/local/java/es/logstash/ 目录下，

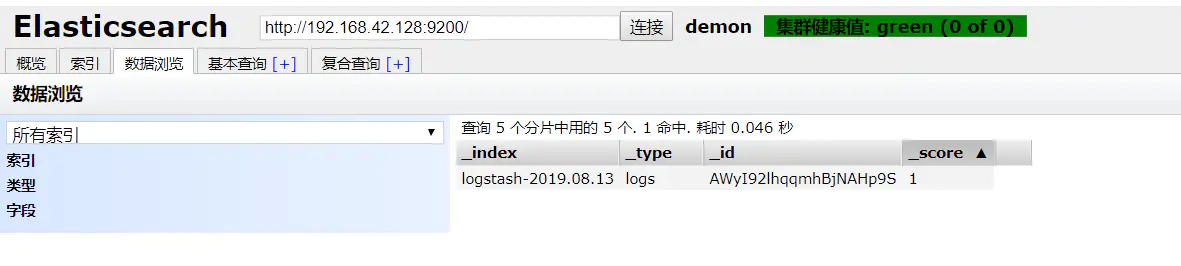
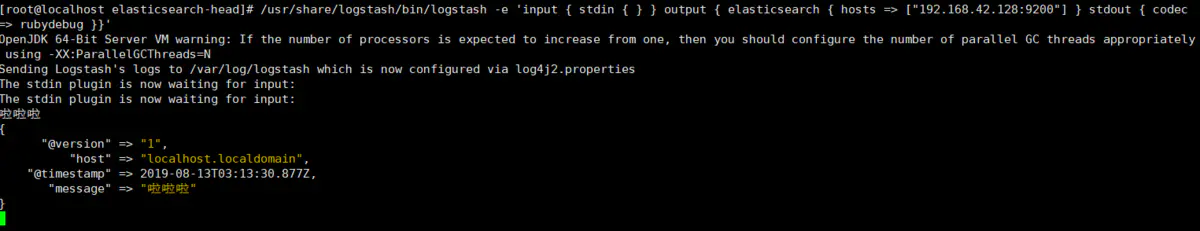
执行命令：./bin/logstash -e 'input { stdin { } } output { stdout {} }'



如果标准输出还有elasticsearch中都需要保留应该怎么玩，看下面

# /usr/local/java/es/logstash/bin/logstash -e 'input { stdin { } } output { elasticsearch { hosts => ["192.168.0.107:9200"] } stdout { codec => rubydebug }}'

运行成功以后输入: 啦啦啦

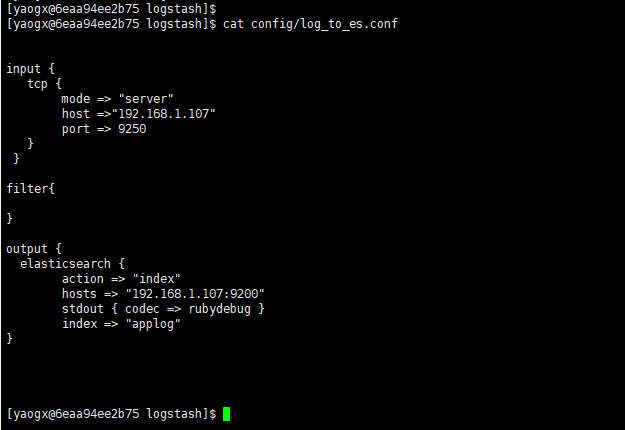


## 修改配置

创建配置文件log\_to\_es.conf

执行命令：vim config/log\_to\_es.conf 新增内容如下：

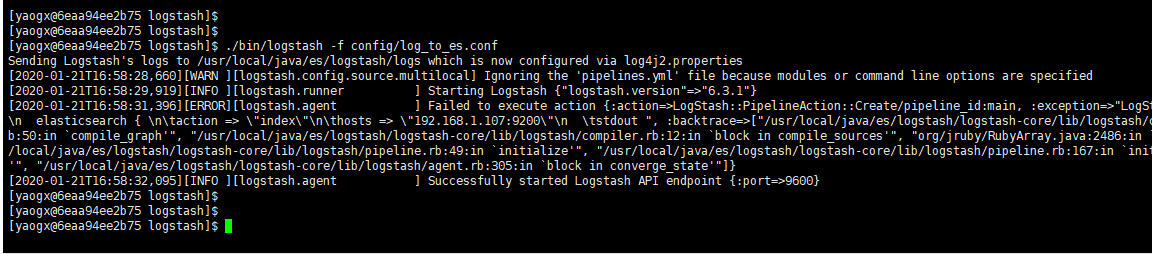
|  |
| --- |
| input {  tcp {  mode => "server"  host =>"192.168.1.107" # 主机 ip  port => 9250 # logstash 端口号  }  }  filter{  }  output {  elasticsearch {  action => "index"  hosts => "192.168.1.107:9200" # elasticsearch 的服务地址  stdout { codec => rubydebug }  index => "applog"  } |



## 启动

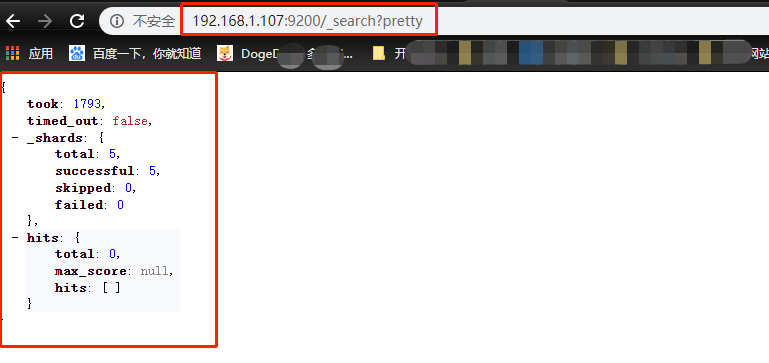
前台启动，执行命令：./bin/logstash -f config/log\_to\_es.conf

后台启动，执行命令：



### 测试是否安装成功

在浏览器中输入：<http://192.168.1.107:9200/_search?pretty>

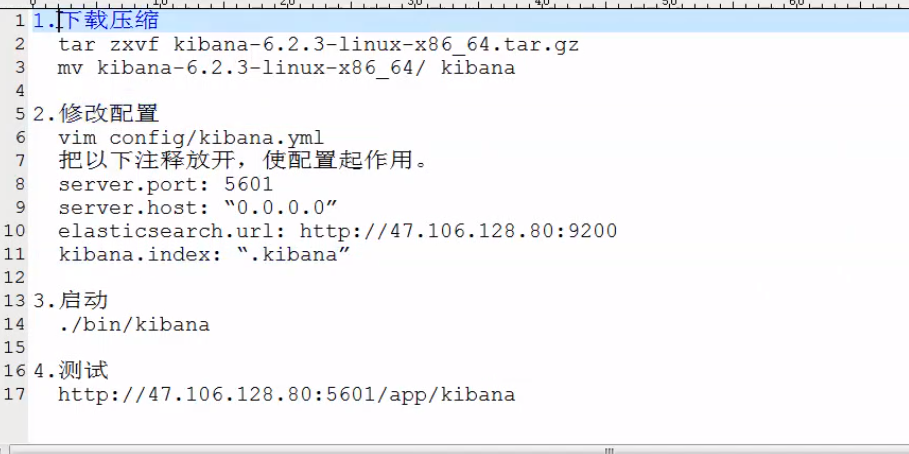


说明：logstash 安装成功。

# 安装Kibana

Kibana 也是一个开源和免费的工具，它Kibana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。

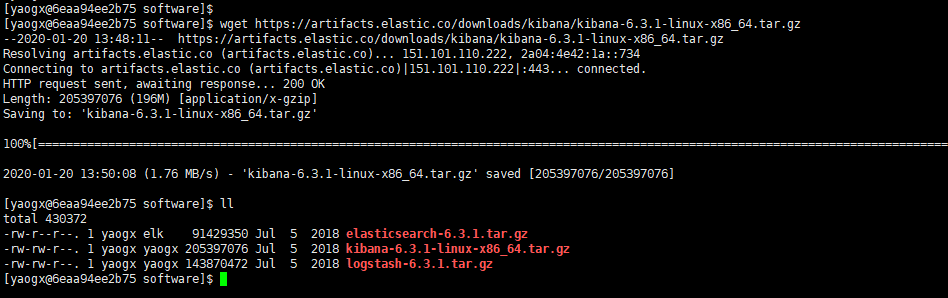
整体流程：



## 下载Kibana

执行命令：

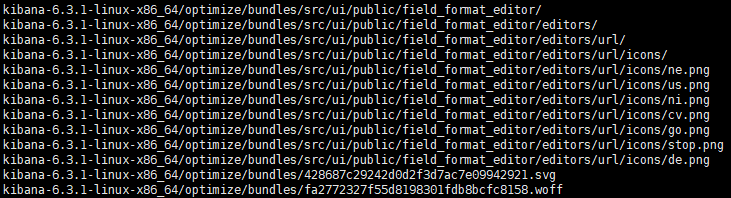
wget <https://artifacts.elastic.co/downloads/kibana/kibana-6.3.1-linux-x86_64.tar.gz>



### 解压Kibana

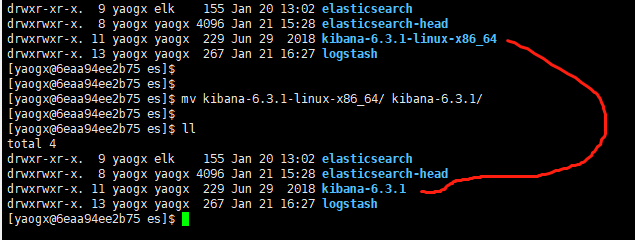
解压Kibana 到指定文件夹，执行命令：

tar -zxvf kibana-6.3.1-linux-x86\_64.tar.gz -C /usr/local/java/es/



### 重命名解压文件

执行命令：mv kibana-6.3.1-linux-x86\_64/ kibana-6.3.1/

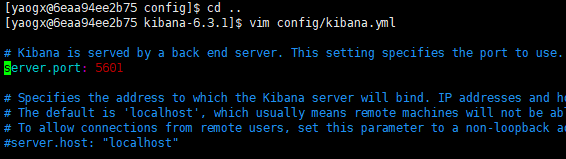


### 修改配置

执行命令：vim config/kibana.yml

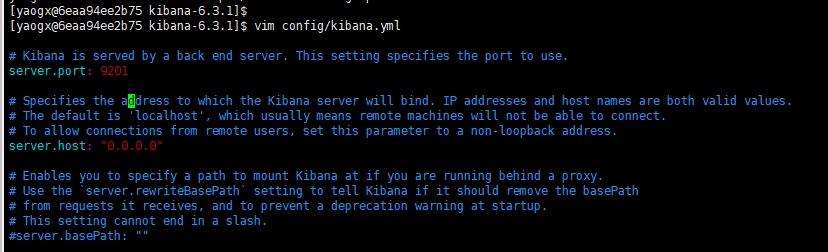
把一下注释放开并修改，使配置起作用。

1. server.port: 5601

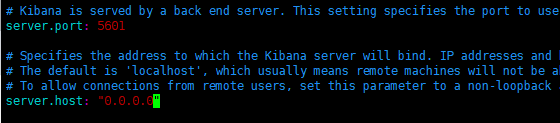


**注：由于我使用的docker 容器，因此需要把端口 设置在【9000 - 9500】之间。**

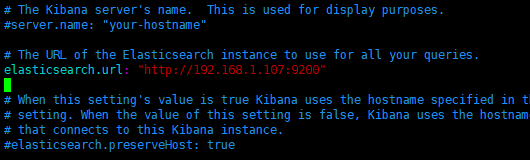
**本 kibana端口设置为：9201**



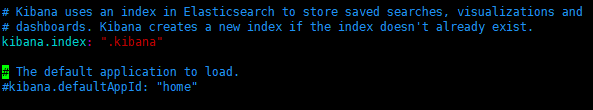
1. server.host: "0.0.0.0"



1. elasticsearch.url: "http://localhost:9200"

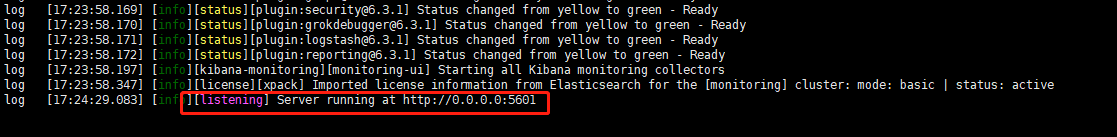
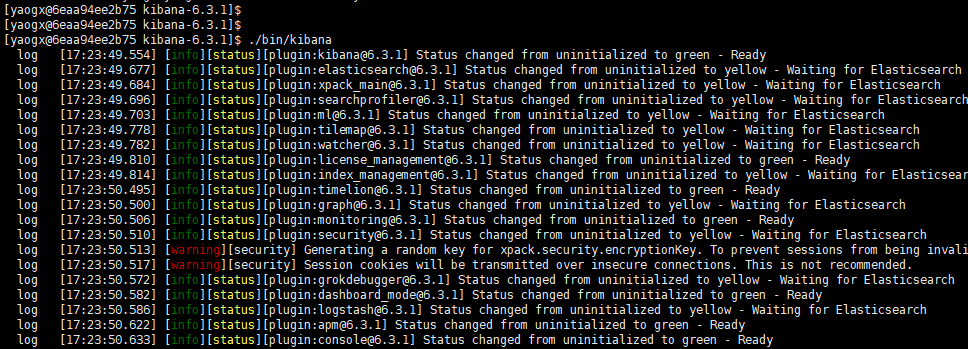


1. kibana.index: ".kibana"



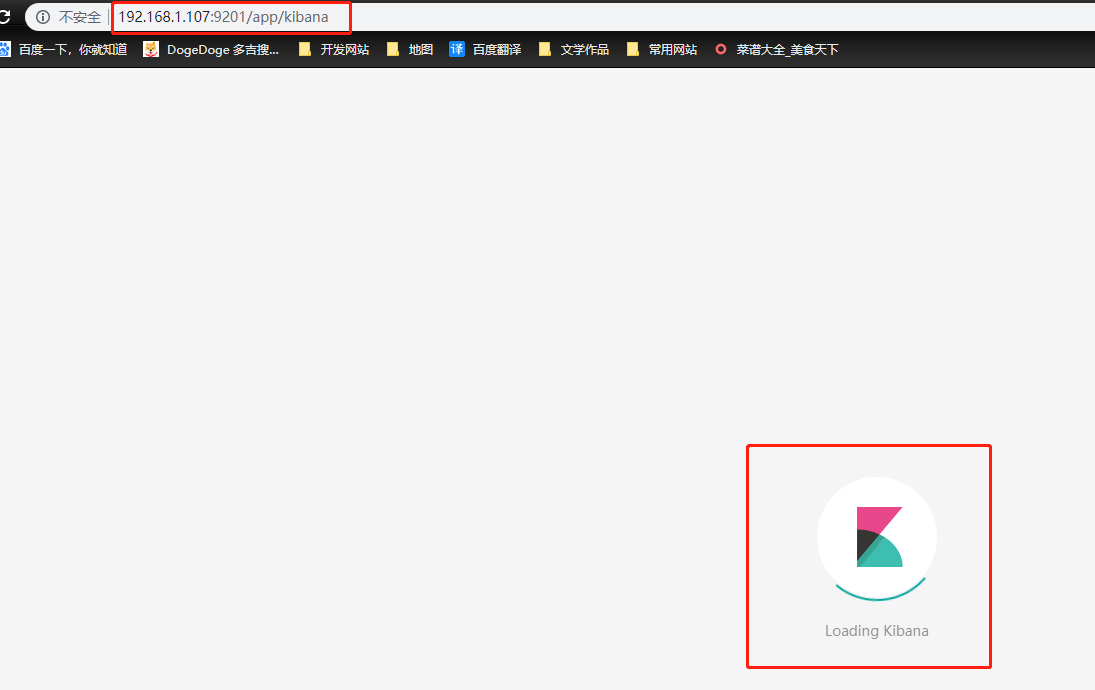
## 启动Kibana

执行命令： ./bin/kibana



如上图表示启动成功。

在浏览器中输入： <http://192.168.1.107:9201/app/kibana>



# 安装nginx

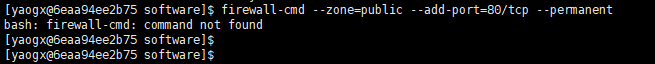
**注：使用root 账户执行如下操作。**

安装nginx 前，需要安装一些软件。

## 防火墙开启端口

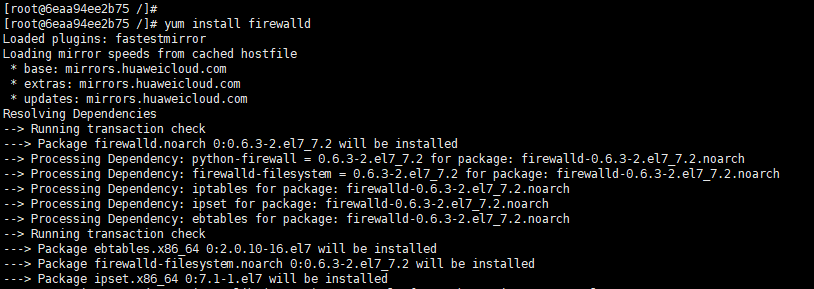
防火墙开启指定 80端口，执行命令：sudo firewall-cmd --zone=public --add-port=80/tcp --permanent

执行命令，如果报错，



解决：通过root 账户安装 firewalld 。

执行命令：yum install firewalld



问题，执行命令：systemctl stop firewalld.service

总是报Failed to get D-Bus connection: Operation not permitted 这样的错误提示。

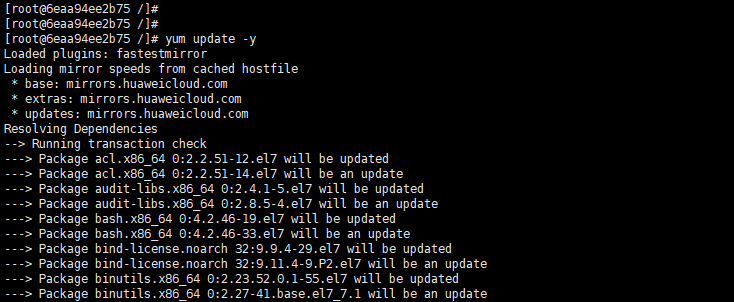
解决办法就是在docker run 的时候运行/usr/sbin/init 。比如：

docker run -tid --name hadoopbase centos/hadoopbase:v001 /usr/sbin/init

## 安装辅助软件

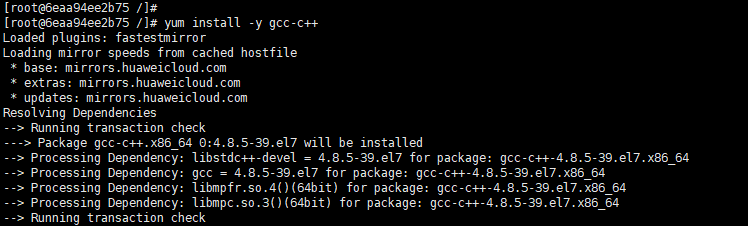
### 更新 yum

执行命令： yum update -y



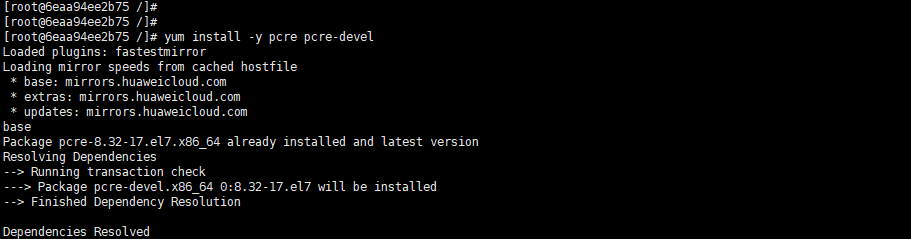
### 安装gcc

执行命令：yum install -y gcc-c++



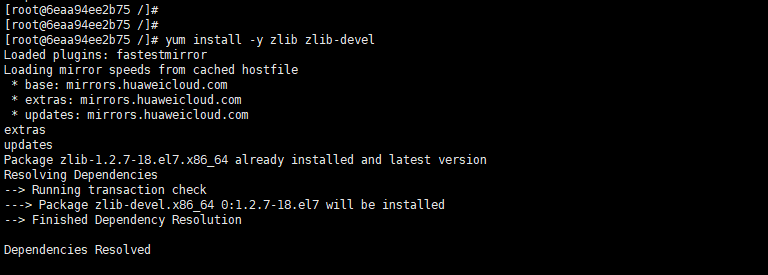
### 安装pcre

执行命令：yum install -y pcre pcre-devel



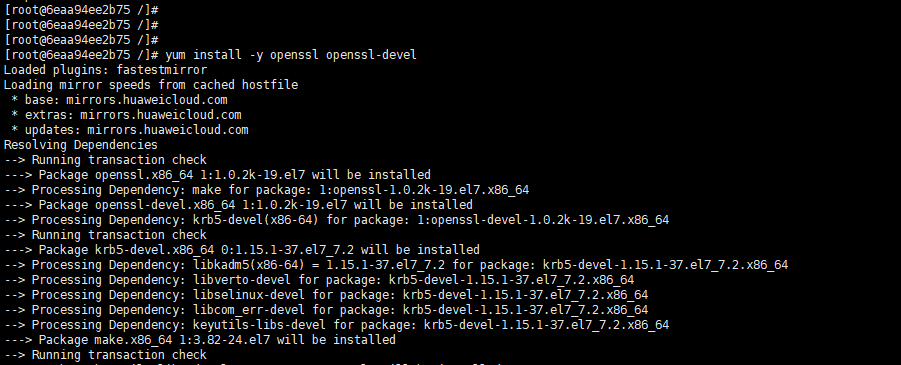
### 安装zlib

执行命令：yum install -y zlib zlib-devel



### 安装OpenSSL

执行命令：yum install -y openssl openssl-devel

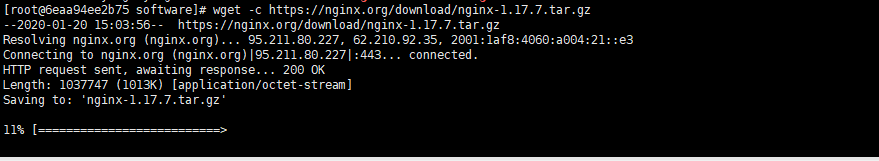


## 下载 nginx

进入 目录：cd /mnt/software/ ；

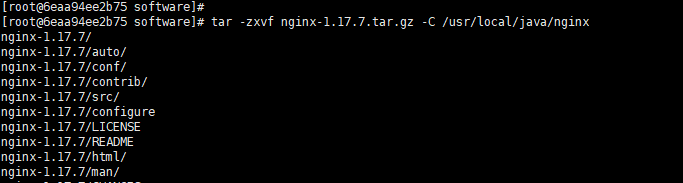
下载nginx1.10.1源码包：

wget -c [https://nginx.org/download/nginx-1.17.7.tar.gz；](https://nginx.org/download/nginx-1.10.1.tar.gz；)



### 解压nginx

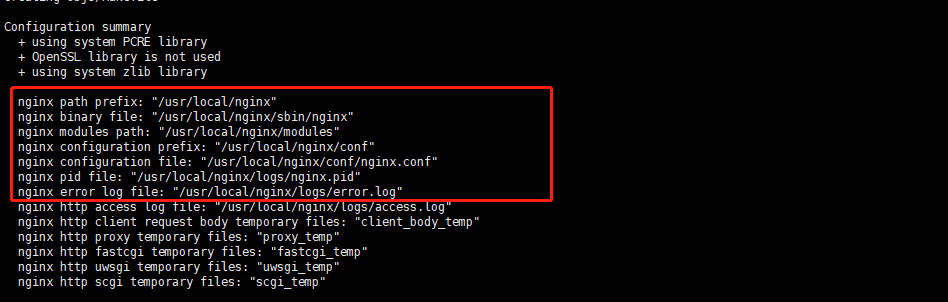
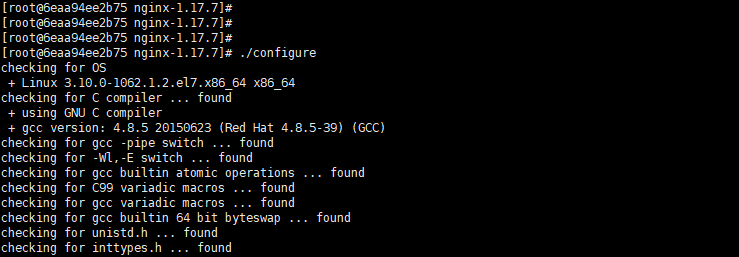
解压到指定目录 ：tar -zxvf nginx-1.17.7.tar.gz -C /usr/local/java/nginx/



进入解压后的nginx 目录：cd /usr/local/java/nginx/nginx-1.17.7；

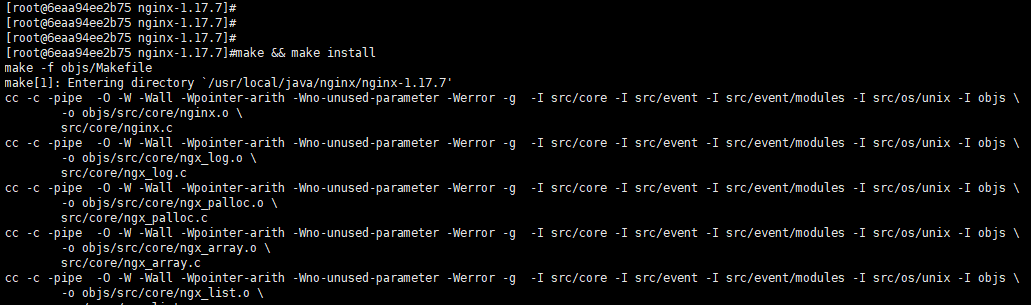
### nginx的配置

使用默认配置,执行命令 ： ./configure



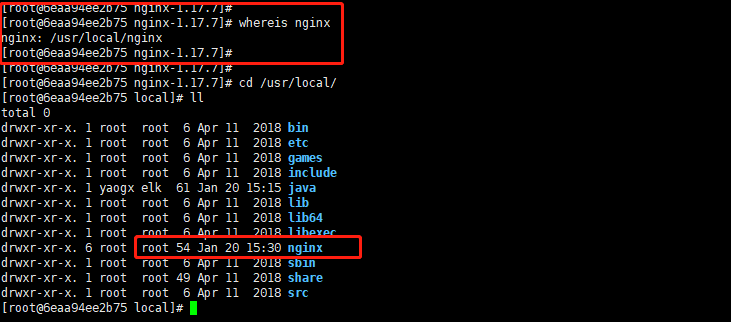
### 编译并安装nginx

执行命令：make && make install



### 查找安装路径

执行命令：whereis nginx

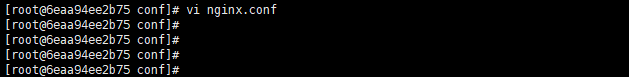


### 修改端口

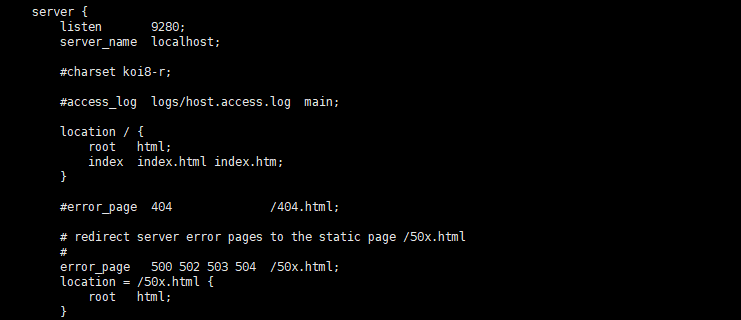
因为该容器与宿主机映射的端口在 【9000-9500】 之间，因此把端口定义在此区间之间。

进入 安装目录： cd /usr/local/nginx/conf/

执行命令：vi nginx.conf



修改 80 端口为：9280



### 启动nginx

**注：启动的是编译安装后的nginx**，非解压后的nginx。

启动nginx： /usr/local/nginx/sbin/nginx；

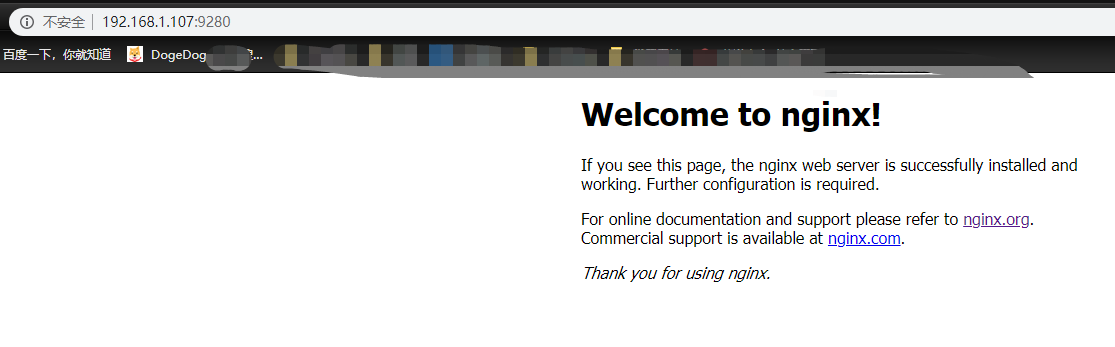
或 进入 cd /usr/local/nginx/sbin/ 目录，执行命令：./nginx

快速停止nginx：/usr/local/nginx/sbin/nginx -s stop；

优雅停止nginx（会将正在处理中的请求处理完毕）：/usr/local/nginx/sbin/nginx -s quit；

## 访问nginx

在浏览器中输入：<http://192.168.1.107:9280/>



# Mysql和ElasticSearch数据同步

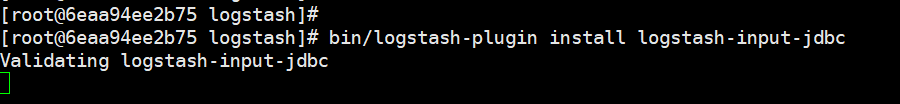
使用logstash进行Mysql和ElasticSearch的同步,首先，将我们刚才给予的下载链接里面的jdbc驱动包上传到容器中.

## 安装插件

logstash-input-jdbc原理：使用 logstash-input-jdbc 插件读取 mysql 的数据，这个插件的工作原理比较简单，就是定时执行一个 sql，然后将 sql 执行的结果写入到流中，增量获取的方式没有通过 binlog 方式同步，而是用一个递增字段作为条件去查询，每次都记录当前查询的位置，由于递增的特性，只需要查询比当前大的记录即可获取这段时间内的全部增量，一般的递增字段有两种，AUTO\_INCREMENT 的主键 id 和 ON UPDATE CURRENT\_TIMESTAMP 的 update\_time 字段，id 字段只适用于那种只有插入没有更新的表，update\_time 更加通用一些，建议在 mysql 表设计的时候都增加一个 update\_time 字段

安装插件用于定时同步数据，执行命令：

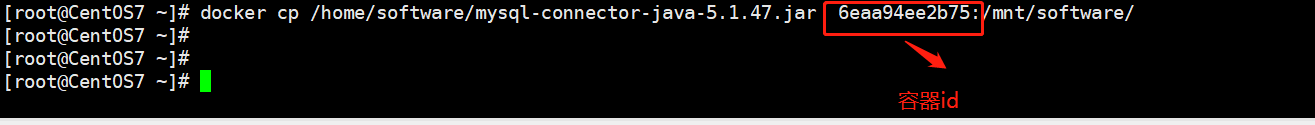
|  |
| --- |
| bin/logstash-plugin install logstash-input-jdbc  bin/logstash-plugin install logstash-output-elasticsearch |



## 从宿主机拷贝文件到容器

执行命令：docker cp 宿主机中要拷贝的文件名及其路径 容器名：要拷贝到容器里面对应的路径 。

Eg: docker cp /home/software/mysql-connector-java-5.1.47.jar 6eaa94ee2b75:/mnt/software/

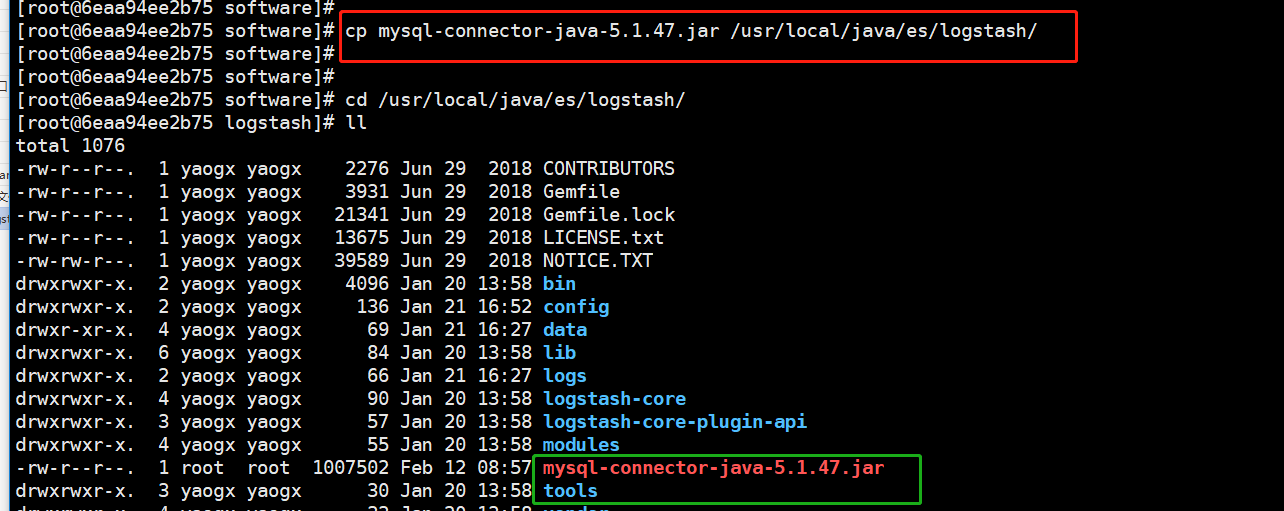


## 复制jar到logstash

首先，将我们刚才给予的下载链接里面的jdbc驱动包放到logstash目录里面来。

执行命令：

cp mysql-connector-java-5.1.47.jar /usr/local/java/es/logstash/



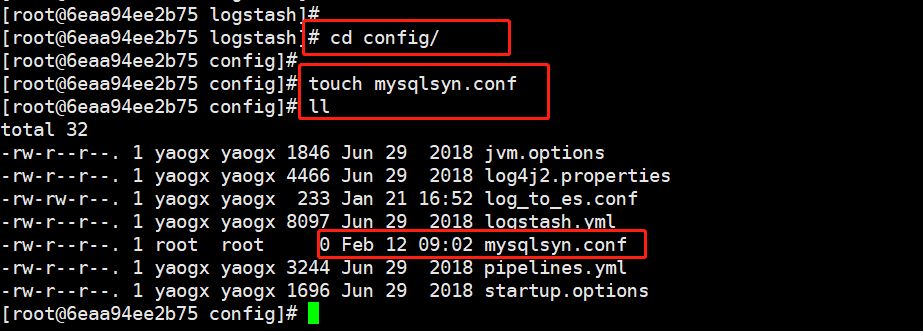
## 配置mysqlsyn.conf文件

由于 logstash 中没有mysqlsyn.conf 文件，所有需要创建该文件。

### 创建mysqlsyn.conf 文件

进入logstash 的config目录中，创建文件mysqlsyn.conf。

执行命令：touch mysqlsyn.conf

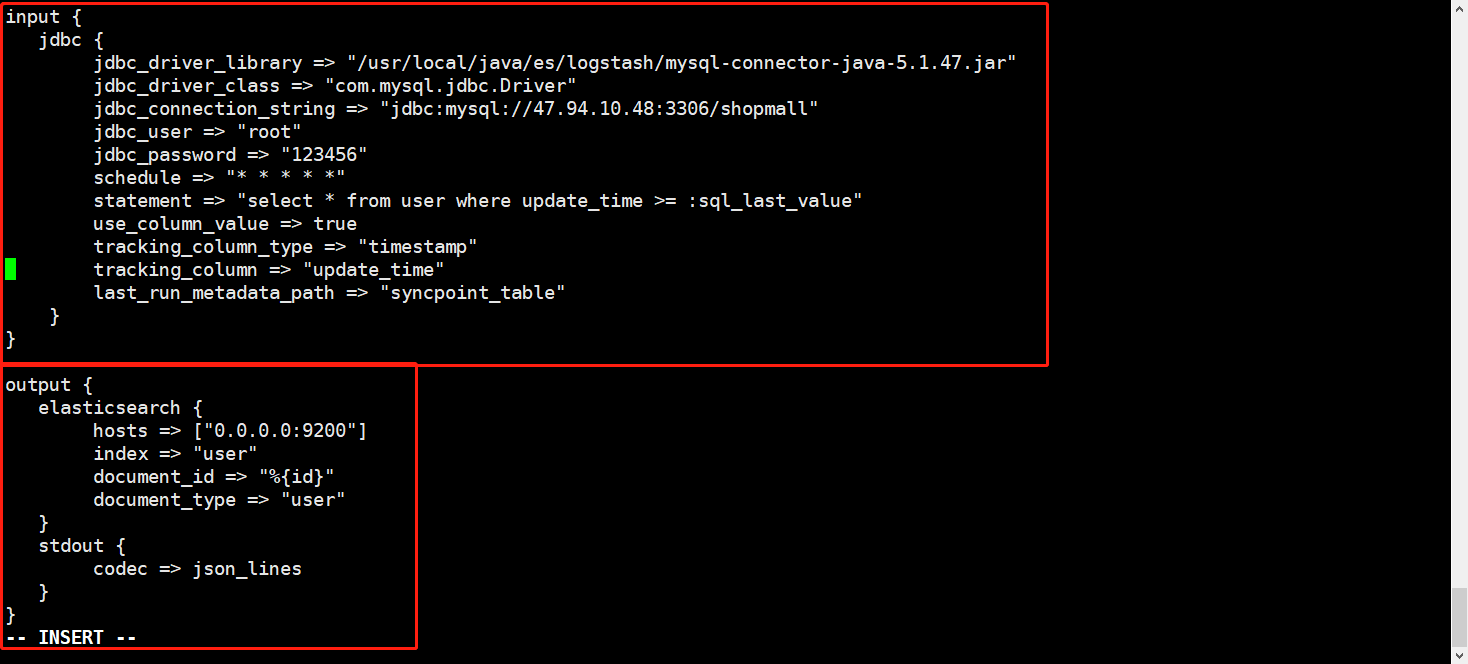
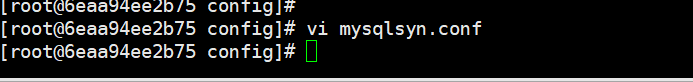


### vim编辑 mysqlsyn.conf 文件

使用vim编辑器打开这个文件，并向里面添加以下内容,并且保存退出。

|  |
| --- |
| input {  jdbc {  # jdbc驱动包位置  jdbc\_driver\_library => "/mnt/logstash-6.6.1/mysql-connector-java-5.1.46/mysql-connector-java-5.1.46-bin.jar"  # 要使用的驱动包类，有过java开发经验的应该很熟悉这个了，不同的数据库调用的类不一样。  jdbc\_driver\_class => "com.mysql.jdbc.Driver"  # myqsl数据库的连接信息  jdbc\_connection\_string => "jdbc:mysql://0.0.0.0:3306/myapp"  # mysql用户  jdbc\_user => "root"  # mysql密码  jdbc\_password => "root"  # 定时任务， 多久执行一次查询, 默认一分钟，如果想要没有延迟，可以使用 schedule => "\* \* \* \* \* \*"  schedule => "\* \* \* \* \*"  # 你要执行的语句  statement => "select \* from user where update\_time >= :sql\_last\_value"  use\_column\_value => true  tracking\_column\_type => "timestamp"  tracking\_column => "update\_time"  last\_run\_metadata\_path => "syncpoint\_table"  }  }  output {  # 将数据输出到ElasticSearch中  elasticsearch {  # es ip加端口  hosts => ["0.0.0.0:9200"]  # es文档索引  index => "myusreinfo"  # es文档数据的id，%{id}代表的是用数据库里面记录的id作为文档的id  document\_id => "%{id}"  document\_type => "user"  }  stdout {  codec => json\_lines  }  } |

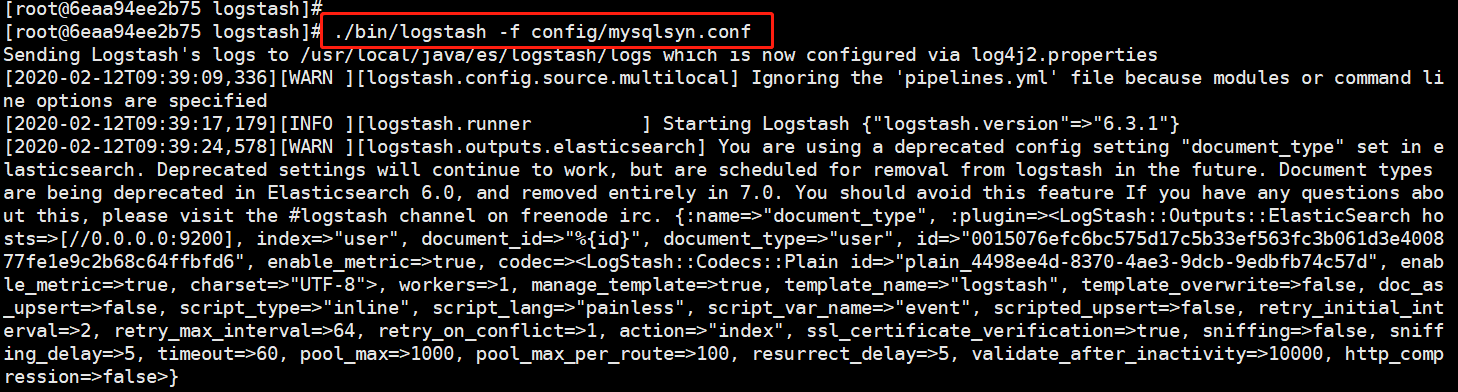
执行命令：vi mysqlsyn.conf



## 启动logstash进行同步

带上配置文件mysqlsyn.conf 启动logstash 。执行命令：

./bin/logstash -f config/mysqlsyn.conf



## 多文件方式同步ES数据

一个 logstash 实例可以借助 pipelines 机制同步多个表，只需要写多个配置文件就可以了，假设我们有两个表 table1 和 table2，对应两个配置文件 sync\_table1.cfg 和 sync\_table2.cfg

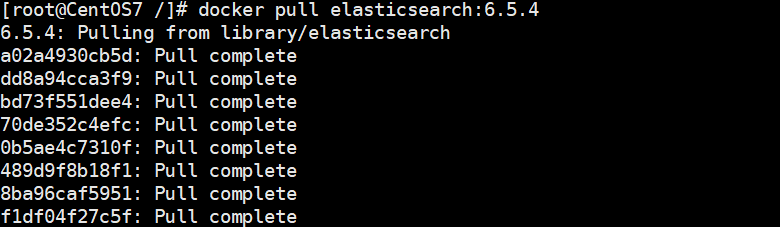
在 config/pipelines.yml 中配置

|  |
| --- |
| - pipeline.id: table1  path.config: "config/mysqlsyn.conf"  - pipeline.id: table2  path.config: "config/mysqlsyn2.conf" |

启动 ，执行命令： ./bin/logstash

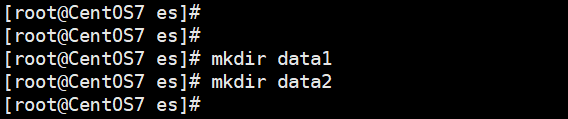
# Docker搭建ELK集群

1. 拉镜像,执行命令：docker pull elasticsearch:6.5.4



1. 在 /usr/local/java/elk 目录下执行以下命令：

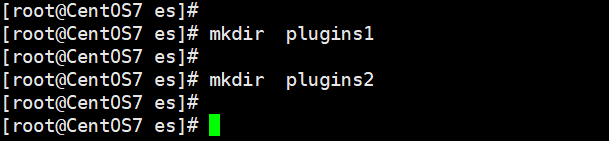
|  |
| --- |
| 1.mkdir -p es/config  2.cd es  3.mkdir data1  4.mkdir data2  firewall-cmd --add-port=9300/tcp  firewall-cmd --add-port=9301/tcp  6.mkdir plugins1  7.mkdir plugins2 |



在防火墙中 开启指定端口。



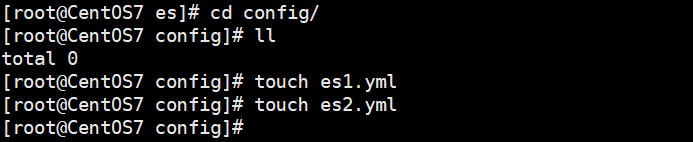
***注：****如果ELK选的6.X版本的，那么读者需将data1 data2 开启777权限=> chmod 777 data1 data2 。*



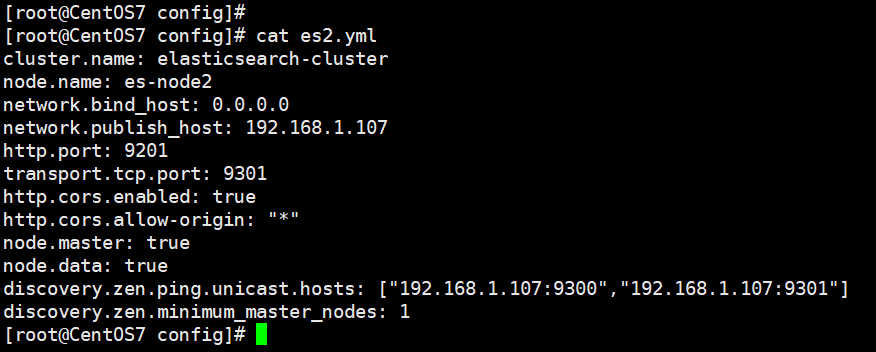
## 配置集群

在es/config分别放入es1.yml、es2.yml

执行命令：touch es1.yml



|  |
| --- |
| 在es/config分别放入es1.yml、es2.yml  Es1:  cluster.name: elasticsearch-cluster  node.name: es-node1  network.bind\_host: 0.0.0.0  network.publish\_host: 192.168.1.107  http.port: 9200  transport.tcp.port: 9300  http.cors.enabled: true  http.cors.allow-origin: "\*"  node.master: true  node.data: true  discovery.zen.ping.unicast.hosts: ["192.168.1.107:9300","192.168.1.107:9301"]  discovery.zen.minimum\_master\_nodes: 1  Es2:  cluster.name: elasticsearch-cluster  node.name: es-node2  network.bind\_host: 0.0.0.0  network.publish\_host: 192.168.1.107  http.port: 9201  transport.tcp.port: 9301  http.cors.enabled: true  http.cors.allow-origin: "\*"  node.master: true  node.data: true  discovery.zen.ping.unicast.hosts: ["192.168.1.107:9300","192.168.1.107:9301"]  discovery.zen.minimum\_master\_nodes: 1 |



## ik中文分词器的安装

到官网下载没有编译的版本自己编译好了进行安装。

官方地址：[https://github.com/medcl/elasticsearch-analysis-ik](https://link.jianshu.com?t=https://github.com/medcl/elasticsearch-analysis-ik" \t "_blank)。

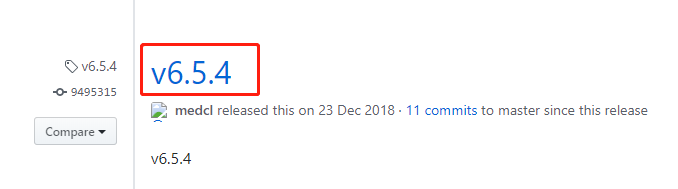
不过这种方法稍微麻烦一点，我们可以用第二种方法：直接下载官方已经编译好的版本。

1. 下载编译好的安装包：

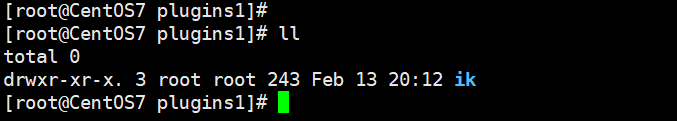
[https://github.com/medcl/elasticsearch-analysis-ik/releases](https://link.jianshu.com?t=https://github.com/medcl/elasticsearch-analysis-ik/releases" \t "_blank)。

**注意:** 下载版本要对应。

本教程下载和elasticsearch:6.5.4 对应的版本。



1. 下载好了之后解压，将解压后的文件夹放在elasticsearch所挂载的目录下plugins目录下，并重命名为ik



## 3.调高JVM线程数限制数量

修改配置sysctl.conf ，执行命令：vim /etc/sysctl.conf

添加如下配置：

vm.max\_map\_count=262144

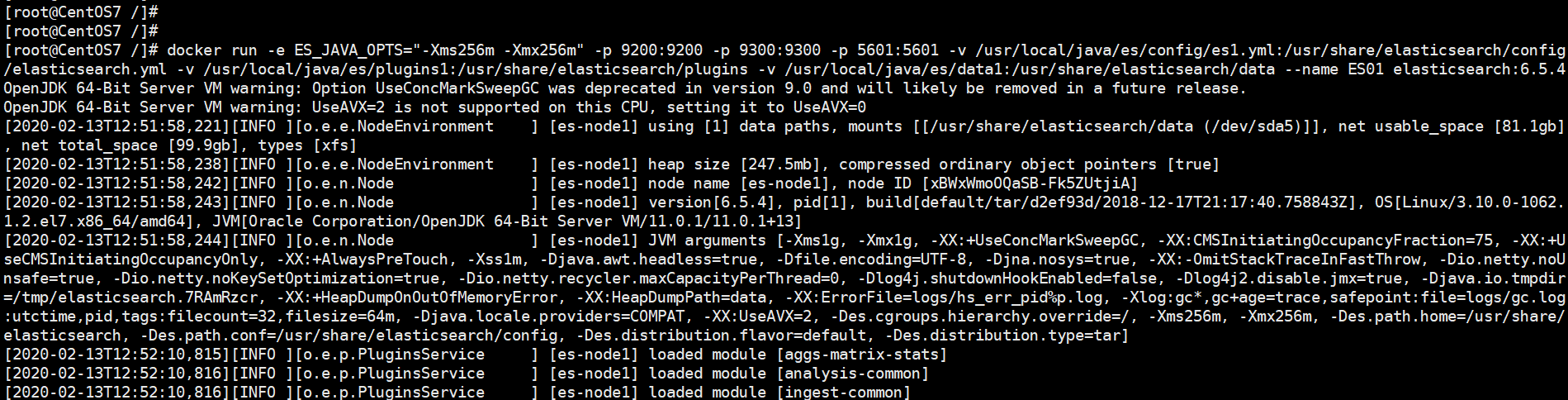
启用配置，执行命令：sysctl -p

## 启动容器

1. 启动第一个elasticsearch

执行命令：

|  |
| --- |
| docker run -e ES\_JAVA\_OPTS="-Xms256m -Xmx256m" -d -p 9200:9200 -p 9300:9300 -p 5601:5601 -v /usr/local/java/elk/es/config/es1.yml:/usr/share/elasticsearch/config/elasticsearch.yml -v /usr/local/java/elk/es/plugins1:/usr/share/elasticsearch/plugins -v /usr/local/java/elk/es/data1:/usr/share/elasticsearch/data --name ES01 elasticsearch:版本号 |



**注：**设置-e ES\_JAVA\_OPTS="-Xms256m -Xmx256m" 是因为

/etc/elasticsearch/jvm.options 默认jvm最大最小内存是2G，读者启动容器后 可用docker stats命令查看

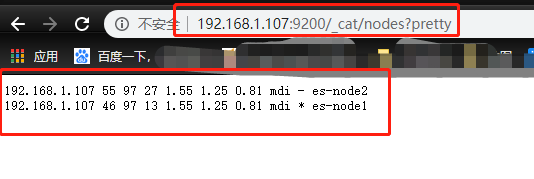
注意：-p 5601:5601 是kibana的端口地址 (我这里kibana的container共用elasticsearch的网络，所以这样设置

1. 启动第二个elasticsearch

|  |
| --- |
| docker run -e ES\_JAVA\_OPTS="-Xms256m -Xmx256m" -d -p 9201:9201 -p 9301:9301 -v /usr/local/java/es/config/es2.yml:/usr/share/elasticsearch/config/elasticsearch.yml -v /usr/local/java/es/plugins2:/usr/share/elasticsearch/plugins -v /usr/local/java/es/data2:/usr/share/elasticsearch/data --name ES02 elasticsearch:6.5.4 |

## 验证是否搭建成功

在浏览器地址栏访问http://192.168.1.107:9200/\_cat/nodes?pretty 查看节点状态

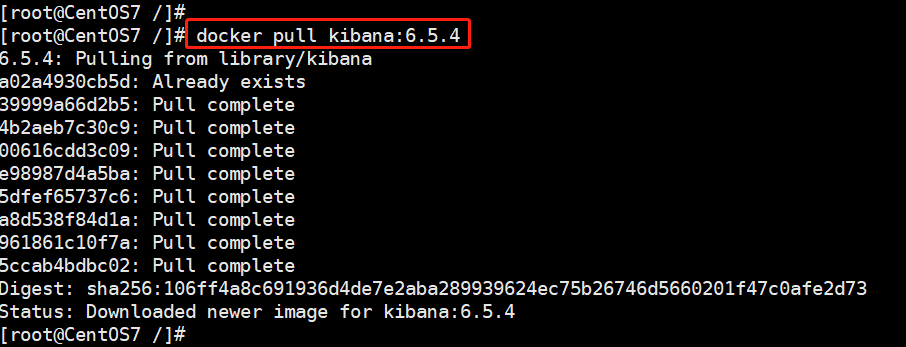


# Kibana的安装

**注意：**kibana **的版本号和**elasticsearch 保持一致。

## 从官网上拉取镜像

拉取kibana 镜像 执行命令： docker pull kibana:6.5.4



## 运行容器

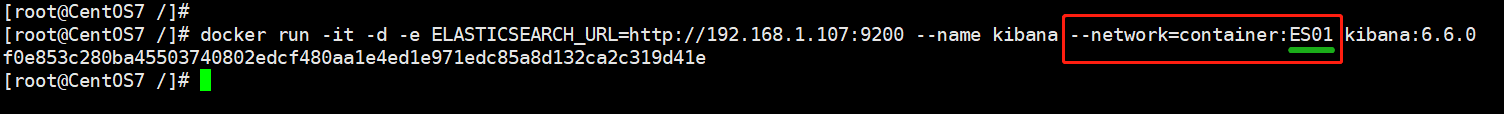
执行命令：

docker run -it -d -e ELASTICSEARCH\_URL=http://192.168.1.107:9200 --name kibana --network=container:ES01 kibana:6.5.4

--network 指定容器共享elasticsearch容器的网络栈 (使用了--network 就不能使用-p 来暴露端口)

ES01 : elasticsearch 集群的容器名 。

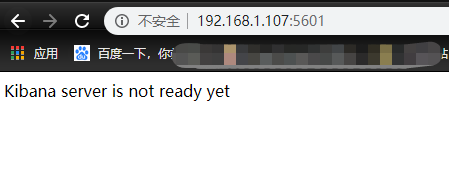
Tag : kibana 的版本号



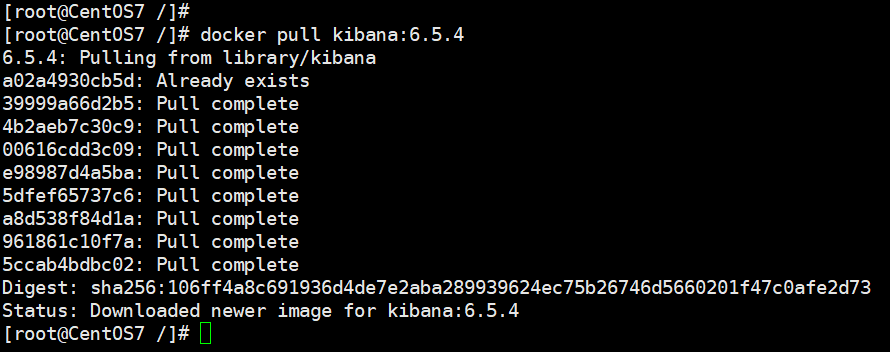
## 访问

在浏览器中输入：<http://192.168.1.107:5601/>

出现如下情况 ，可能是 elasticsearch 和kibana 的版本不一致导致。



解决：通过docker 命令删除kibana ，重新拉取和elasticsearch 相同的版本号的kibana 。

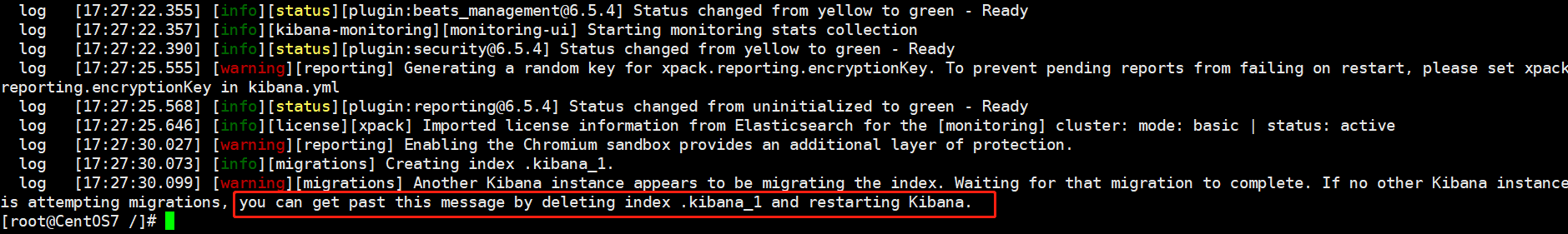


重新按照上边流程安装，不再赘述。

## 安装过程遇到的错误

错误一： log [17:27:30.099] [warning][migrations] Another Kibana instance appears to be migrating the index. Waiting for that migration to complete. If no other Kibana instance is attempting migrations, you can get past this message by deleting index .kibana\_1 and restarting Kibana.

另一个Kibana实例似乎正在迁移索引。等待该迁移完成。如果没有其他Kibana实例正在尝试迁移，则可以通过删除索引.kibana\_1和重新启动Kibana。



解决：执行命令：curl -XDELETE [http：// localhost：9200 / .kibana\_1](http://localhost:9200/.kibana_1)。

然后重启 Kibana 。

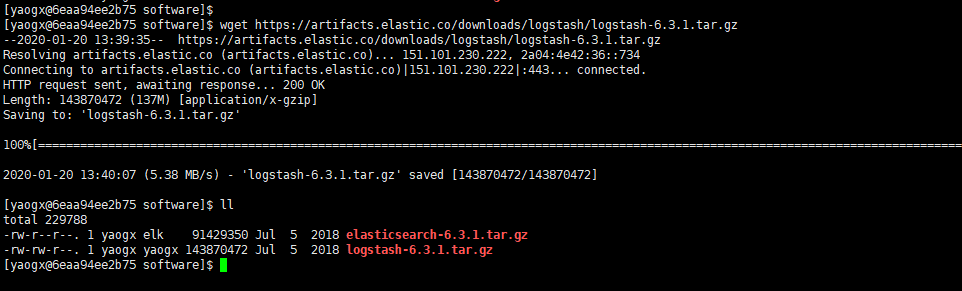
# 安装logstash

Logstash是一个完全开源的工具，他可以对你的日志进行收集、过滤，并将其存储供以后使用（如，搜索）。

## 下载logstash

执行命令：

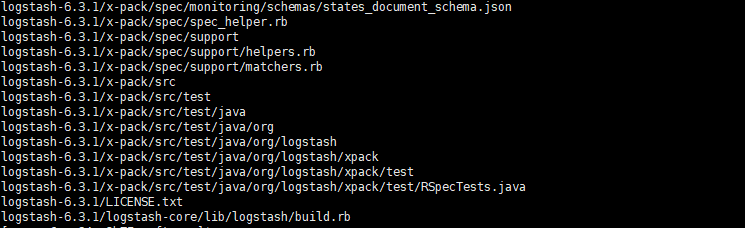
wget https://artifacts.elastic.co/downloads/logstash/logstash-6.3.1.tar.gz



## 解压logstash

解压到指定文件夹： /usr/local/java/elk/

执行命令：tar -zxvf logstash-6.3.1.tar.gz -C /usr/local/java/elk/

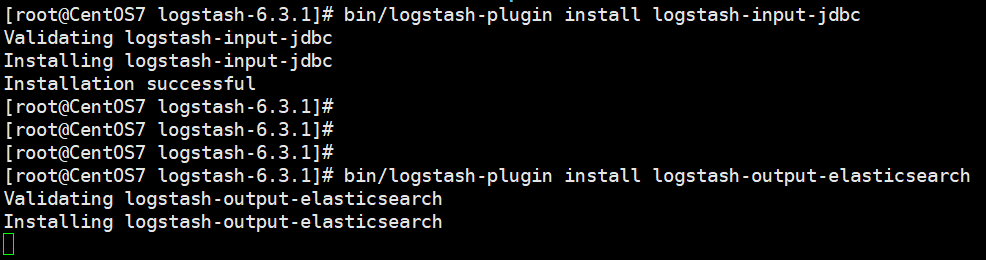


## logstash 同步mysql 数据到 elasticsearch

进入 logstash-6.3.1 目录，然后执行下列命令。

执行命令;

|  |
| --- |
| bin/logstash-plugin install logstash-input-jdbc  bin/logstash-plugin install logstash-output-elasticsearch |



### 复制jar到logstash

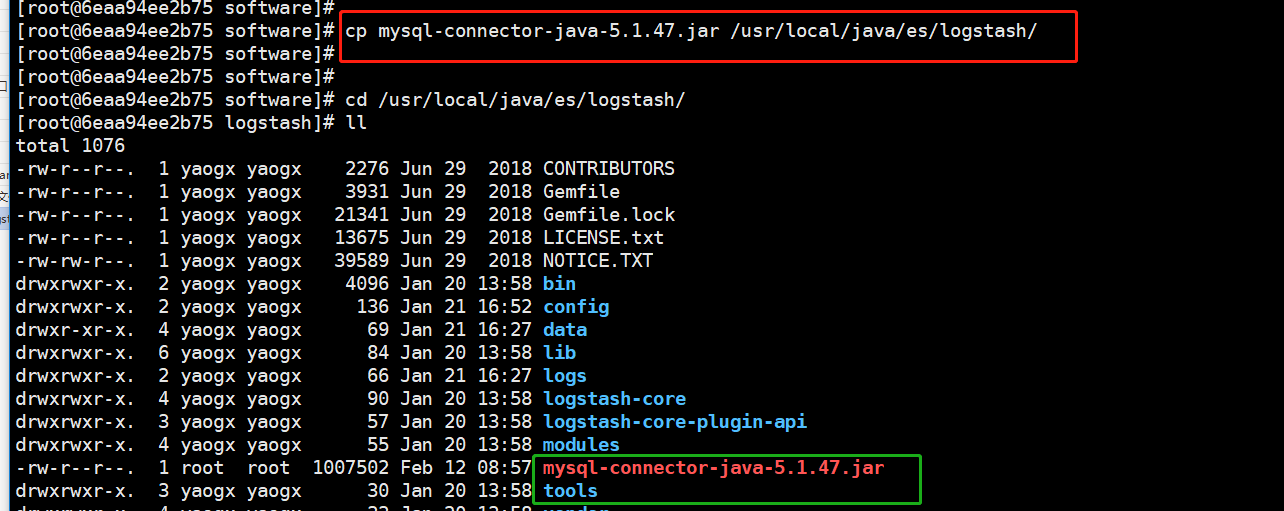
Logstash 同步mysql数据需要jdbc驱动包

mysql-connector-java-5.1.47.jar ，因此上传驱动包

mysql-connector-java-5.1.47.jar 放到logstash目录里面来。

执行命令：

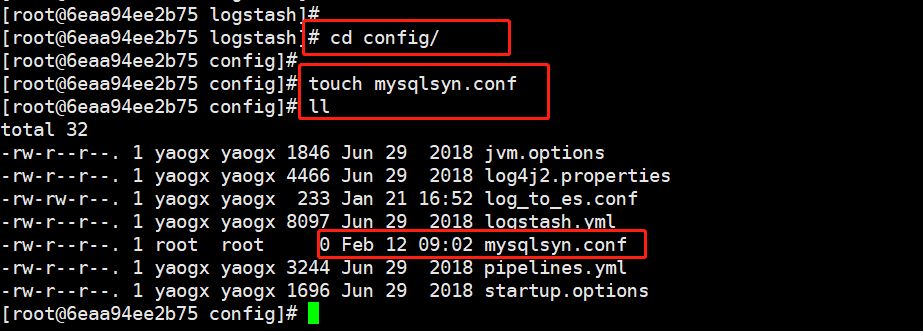
cp mysql-connector-java-5.1.47.jar /usr/local/java/elk/logstash/



### 创建mysqlsyn.conf 文件

进入logstash 的config目录中，创建文件mysqlsyn.conf。

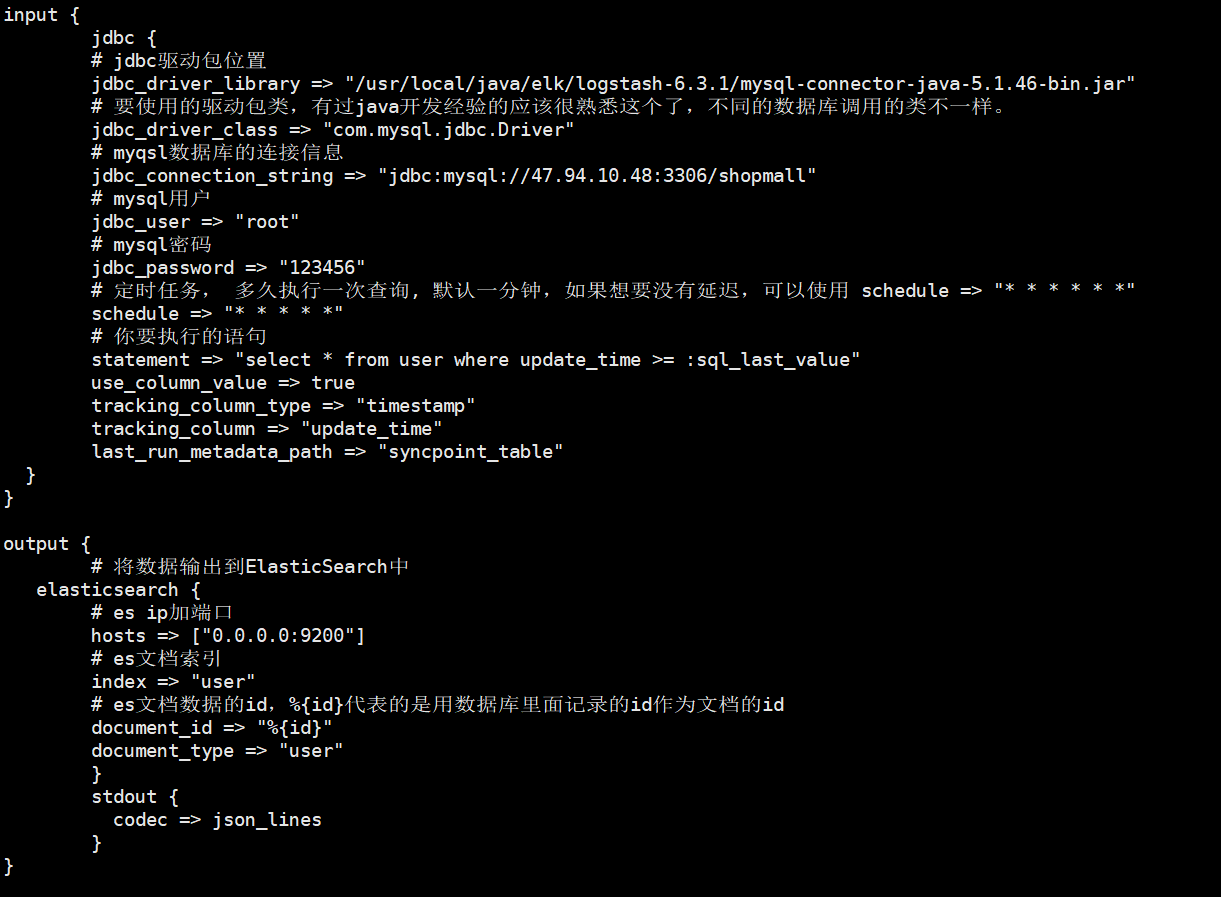
执行命令：touch mysqlsyn.conf



### vim编辑 mysqlsyn.conf 文件

使用vim编辑器打开这个文件，并向里面添加以下内容,并且保存退出。

|  |
| --- |
| input {  jdbc {  # jdbc驱动包位置  jdbc\_driver\_library => "/mnt/logstash-6.6.1/mysql-connector-java-5.1.46/mysql-connector-java-5.1.46-bin.jar"  # 要使用的驱动包类，有过java开发经验的应该很熟悉这个了，不同的数据库调用的类不一样。  jdbc\_driver\_class => "com.mysql.jdbc.Driver"  # myqsl数据库的连接信息  jdbc\_connection\_string => "jdbc:mysql://0.0.0.0:3306/myapp"  # mysql用户  jdbc\_user => "root"  # mysql密码  jdbc\_password => "root"  # 定时任务， 多久执行一次查询, 默认一分钟，如果想要没有延迟，可以使用 schedule => "\* \* \* \* \* \*"  schedule => "\* \* \* \* \*"  # 你要执行的语句  statement => "select \* from user where update\_time >= :sql\_last\_value"  use\_column\_value => true  tracking\_column\_type => "timestamp"  tracking\_column => "update\_time"  last\_run\_metadata\_path => "syncpoint\_table"  }  }  output {  # 将数据输出到ElasticSearch中  elasticsearch {  # es ip加端口  hosts => ["0.0.0.0:9200"]  # es文档索引  index => "myusreinfo"  # es文档数据的id，%{id}代表的是用数据库里面记录的id作为文档的id  document\_id => "%{id}"  document\_type => "user"  }  stdout {  codec => json\_lines  }  } |



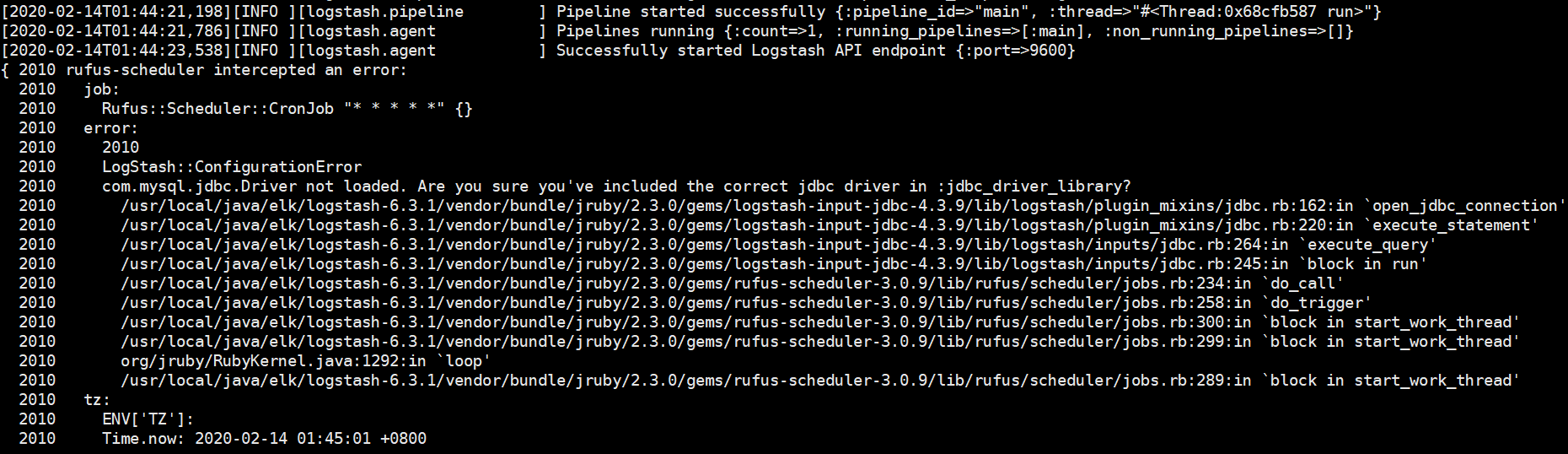
## 启动logstash进行同步

进入logstash 目录中， 带上配置文件mysqlsyn.conf 启动logstash 。执行命令：

./bin/logstash -f config/mysqlsyn.conf

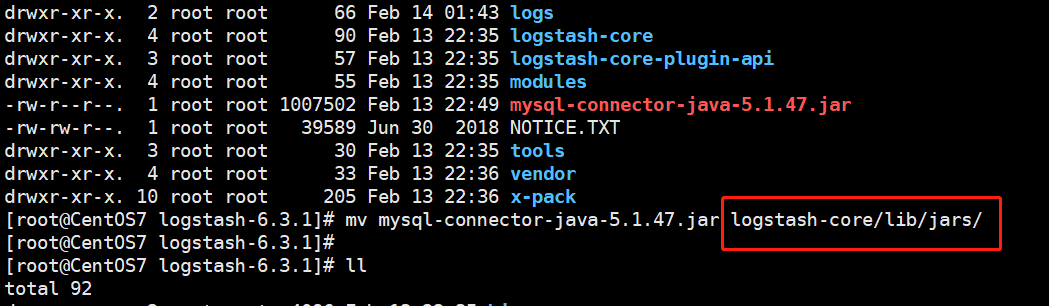
### 启动报错

错误1： com.mysql.jdbc.Driver not loaded. Are you sure you've included the correct jdbc driver in :jdbc\_driver\_library?

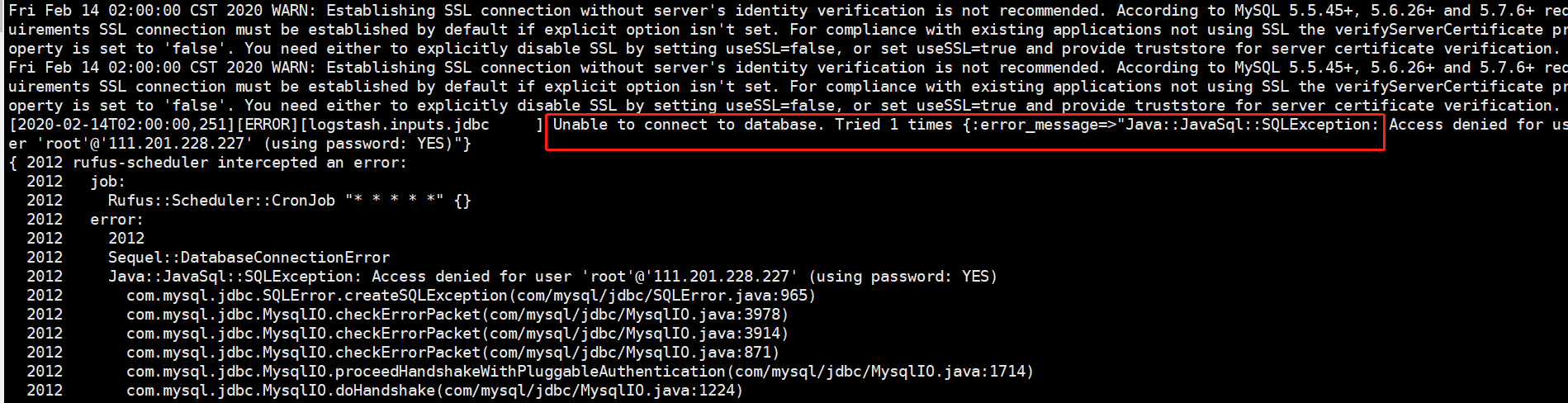


对于Logstash 6.2.x及更高版本，建议mysql连接驱动放在

logstash-6.3.1/logstash-core/lib/jars 目录下。



问题2：Unable to connect to database. Tried 1 times



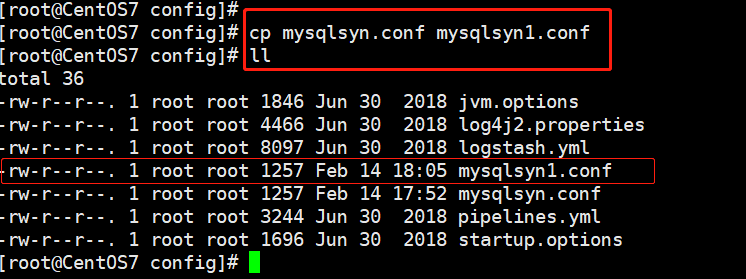
原因：无法连接数据库。

解决：检查数据库连接是否正常，及密码是否正确。

## 同步多表或多库

进入logstash 目录中， 复制mysqlsyn.conf 文件，重新命名：

Mysqlsyn1.conf 。



修改 mysqlsyn1.conf文件 ,修改内容如下：

1. jdbc\_connection\_string ：对应的数据库名称
2. statement => ：对应 要执行查询某个表的语句
3. index => ： 对应 的es 索引。
4. document\_type => "user" ：对应的文档类型，会在实体类中使用。



### 配置 pipelines.yml 文件

一个 logstash 实例可以借助 pipelines 机制同步多个表，只需要写多个配置文件就可以了，假设我们有两个表 table1 和 table2，对应两个配置文件 sync\_table1.cfg 和 sync\_table2.cfg

在 config/pipelines.yml 中配置

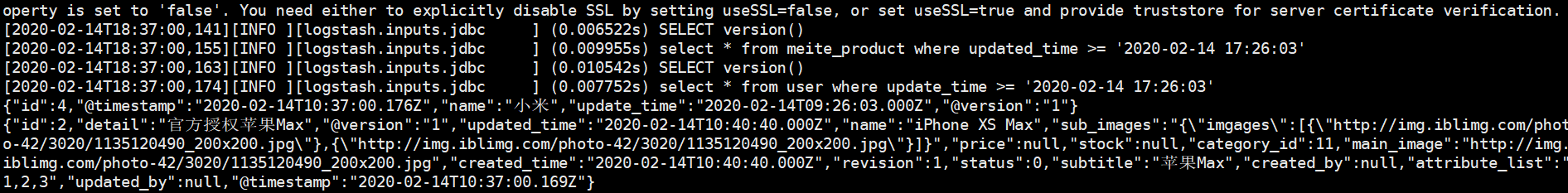
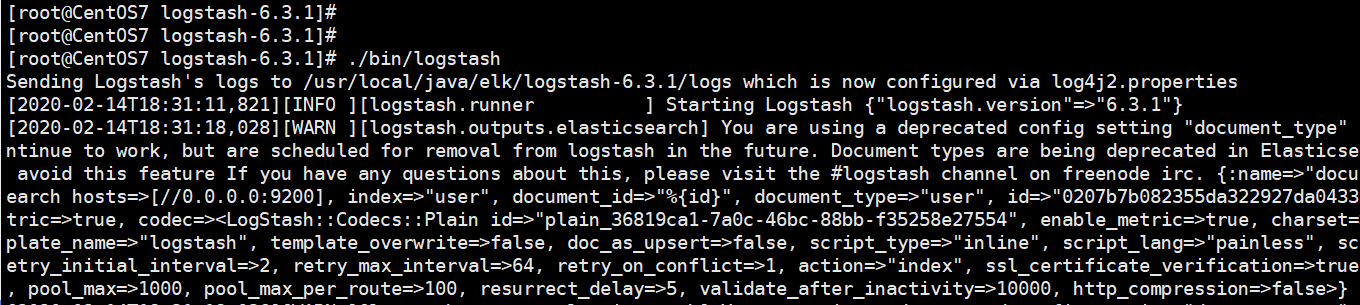
|  |
| --- |
| - pipeline.id: table1  path.config: "config/sync\_table1.cfg"  - pipeline.id: table2  path.config: "config/sync\_table2.cfg" |

配置 pipelines.yml 文件，用于加载 上面两个mysql 同步文件。



### 启动 logstash

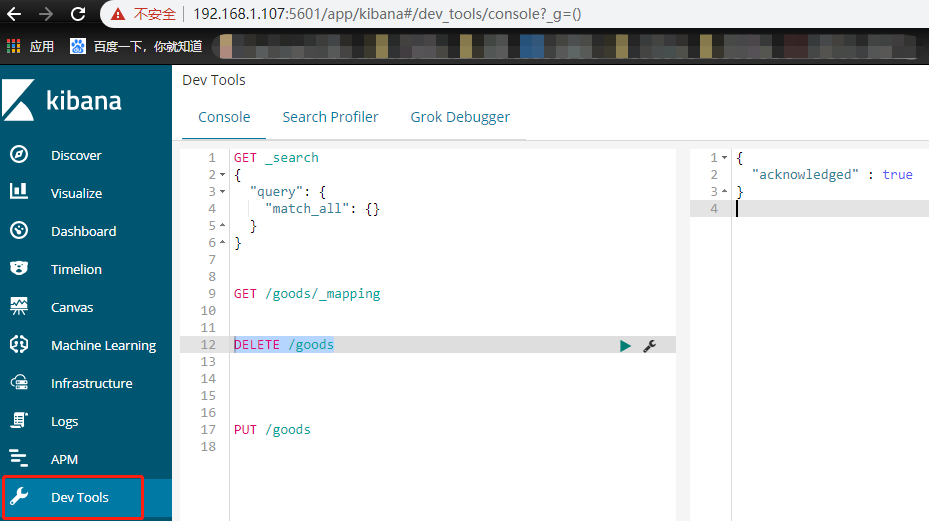
执行命令： ./bin/logstash



## 解决已安装中文分词器项目无法查询中文

**注：再点击三角执行按钮后 ，稍等片刻，等执行完再执行下一条命令。**

以下操作 均在 Kibana 中执行。

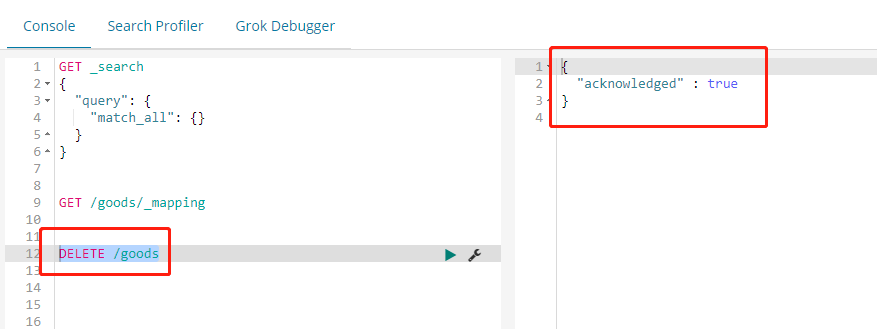


## 查询索引映射

执行命令：GET /goods/\_mapping

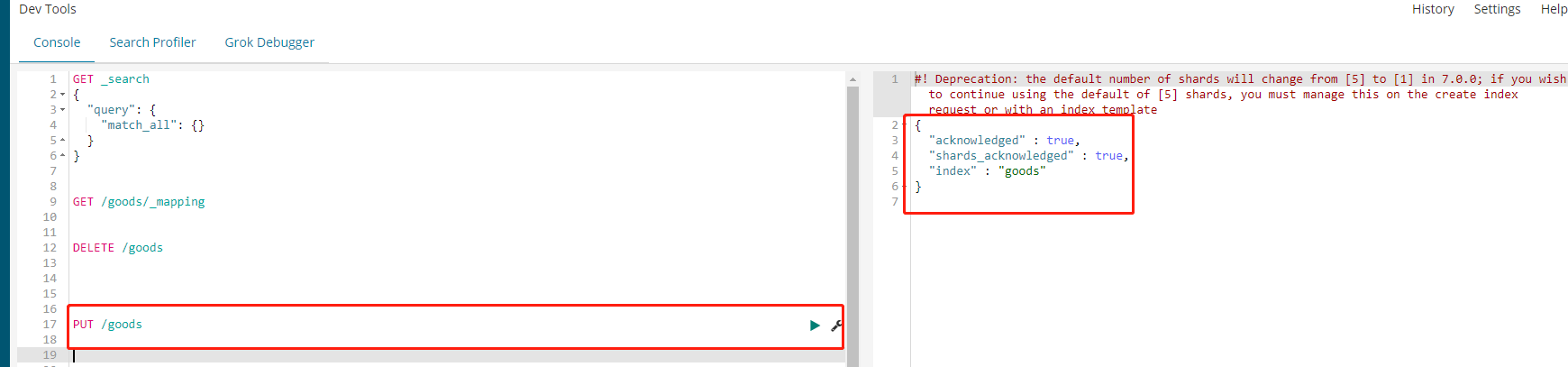
## 删除索引

执行命令：DELETE /goods



## 重新添加索引

执行命令：PUT /goods



## 添加ik分词器

执行命令：

|  |
| --- |
| PUT goods/\_mapping/goods  {  "properties": {  "@timestamp": {  "type": "date"  },  "@version": {  "type": "text",  "fields": {  "keyword": {  "type": "keyword",  "ignore\_above": 256  }  }  },  "attribute\_list": {  "type": "text",  "fields": {  "keyword": {  "type": "keyword",  "ignore\_above": 256  }  }  },  "category\_id": {  "type": "long"  },  "created\_time": {  "type": "date"  },  "detail": {  "type": "text",  "analyzer":"ik\_smart",  "search\_analyzer":"ik\_smart"  },  "id": {  "type": "long"  },  "main\_image": {  "type": "text",  "fields": {  "keyword": {  "type": "keyword",  "ignore\_above": 256  }  }  },  "name": {  "type": "text",  "analyzer":"ik\_smart",  "search\_analyzer":"ik\_smart"  },  "revision": {  "type": "long"  },  "status": {  "type": "long"  },  "sub\_images": {  "type": "text",  "fields": {  "keyword": {  "type": "keyword",  "ignore\_above": 256  }  }  },  "subtitle": {  "type": "text",  "analyzer":"ik\_smart",  "search\_analyzer":"ik\_smart"  },  "updated\_time": {  "type": "date"  }  }  } |

