

1 Elasticsearch 安装

1.1 Elasticsearch安装

1、上传ElasticSearch安装包

```
alt+p # 打开sftp窗口
# 上传es安装包
put e:/software/elasticsearch-7.4.0-linux-x86_64.tar.gz
```

```
192.168.149.135 | SFTP-192.168.149.135
sftp> put e:/software/elasticsearch-7.4.0-linux-x86_64.tar.gz
Uploading elasticsearch-7.4.0-linux-x86_64.tar.gz to /root/elasticsearch-7.4.0-linux-x86_64.tar.gz
100% 284790KB 16752KB/s 00:00:17
e:/software/elasticsearch-7.4.0-linux-x86_64.tar.gz: 291625299 bytes transferred in 17 seconds (16752 KB/s)
```

2、执行解压操作，如下图

```
# 将elasticsearch-7.4.0-linux-x86_64.tar.gz解压到opt文件夹下。 -C 大写
tar -zxvf elasticsearch-7.4.0-linux-x86_64.tar.gz -C /opt
```

3、创建普通用户

因为安全问题，Elasticsearch 不允许root用户直接运行，所以要创建新用户，在root用户中创建新用户,执行如下命令：

```
useradd itheima # 新增itheima用户
passwd itheima # 为itheima用户设置密码
```

5、为新用户授权，如下图

```
chown -R itheima:itheima /opt/elasticsearch-7.4.0 #文件夹所有者
```

```
drwxr-xr-x.  2 itheima itheima  4096 Sep 27 01:40 bin
drwxr-xr-x.  2 itheima itheima   178 Nov 24 07:01 config
drwxr-xr-x. 10 itheima itheima   119 Sep 27 01:40 jdk
drwxr-xr-x.  3 itheima itheima  4096 Sep 27 01:40 lib
-rw-r--r--.  1 itheima itheima 13675 Sep 27 01:35 LICENSE.txt
drwxr-xr-x.  2 itheima itheima  4096 Nov 24 07:01 logs
drwxr-xr-x. 37 itheima itheima  4096 Sep 27 01:40 modules
-rw-r--r--.  1 itheima itheima 523209 Sep 27 01:40 NOTICE.txt
drwxr-xr-x.  2 itheima itheima     6 Sep 27 01:40 plugins
-rw-r--r--.  1 itheima itheima  8500 Sep 27 01:35 README.textile
```

将 /opt/elasticsearch-7.4.0文件夹授权给itheima用户，由上图可见，我们的文件夹权限赋给了itheima

6、修改elasticsearch.yml文件

```
vim /opt/elasticsearch-7.4.0/config/elasticsearch.yml
```

```
# ===== Elasticsearch Configuration =====
cluster.name: my-application
node.name: node-1
network.host: 0.0.0.0
http.port: 9200
cluster.initial_master_nodes: ["node-1"]
```

cluster.name: 配置elasticsearch的集群名称，默认是elasticsearch。建议修改成一个有意义的名称

node.name: 节点名，elasticsearch会默认随机指定一个名字，建议指定一个有意义的名称，方便管理

network.host: 设置为0.0.0.0允许外网访问

http.port: Elasticsearch的http访问端口

cluster.initial_master_nodes: 初始化新的集群时需要此配置来选举master

7、修改配置文件

新创建的itheima用户最大可创建文件数太小，最大虚拟内存太小，切换到root用户，编辑下列配置文件，添加类似如下内容

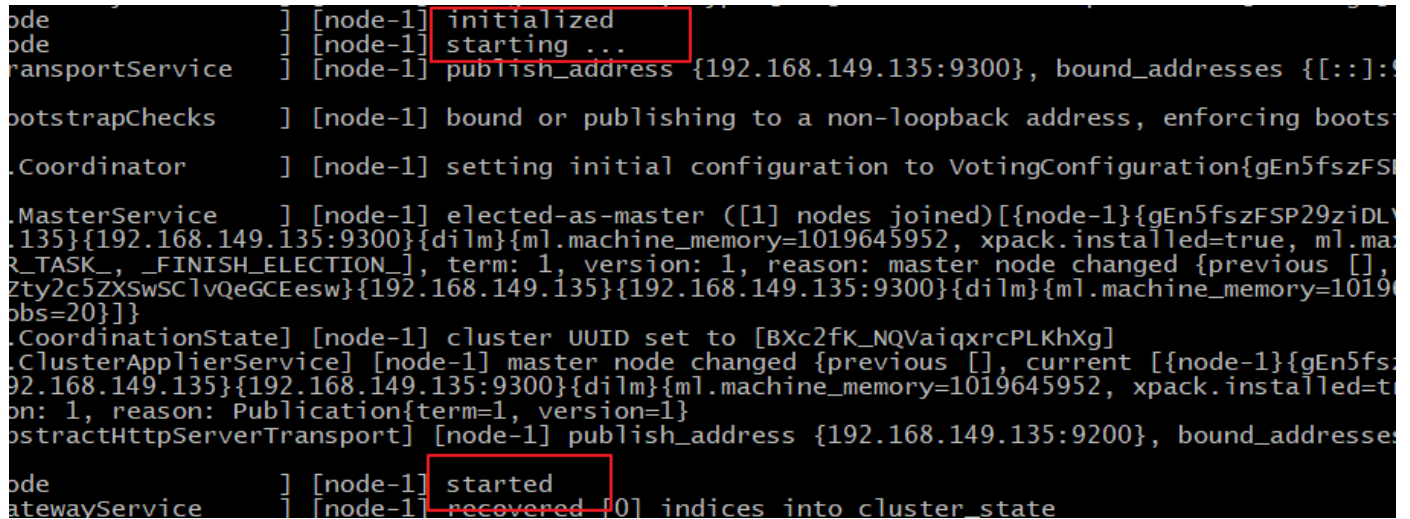
```
# 切换到root用户
su root

#1. ===最大可创建文件数太小=====
vim /etc/security/limits.conf
# 在文件末尾中增加下面内容
itheima soft nofile 65536
itheima hard nofile 65536
# =====
vim /etc/security/limits.d/20-nproc.conf
# 在文件末尾中增加下面内容
itheima soft nofile 65536
itheima hard nofile 65536
* hard nproc 4096
# 注：* 代表Linux所有用户名称

#2. ===最大虚拟内存太小=====
vim /etc/sysctl.conf
# 在文件中增加下面内容
vm.max_map_count=655360
# 重新加载，输入下面命令：
sysctl -p
```

8、启动elasticsearch

```
su itheima # 切换到itheima用户启动
cd /opt/elasticsearch-7.4.0/bin
./elasticsearch #启动
```



The screenshot shows the Elasticsearch startup process. Key log entries include: [node-1] initialized, [node-1] starting ..., [node-1] publish_address {192.168.149.135:9300}, [node-1] bound or publishing to a non-loopback address, [node-1] setting initial configuration to VotingConfiguration{gEn5fszFSP29ziDLV}, [node-1] elected-as-master ([1] nodes joined), [node-1] master node changed {previous [], current [{node-1}{gEn5fszFSP29ziDLV}], term: 1, version: 1, reason: master node changed}, [node-1] cluster UUID set to [BXc2fK_NQVaiqsrcPLKhXg], [node-1] started, and [node-1] recovered [0] indices into cluster state. The log also shows the master node's configuration, including machine memory and xpack installation status.

通过上图我们可以看到elasticsearch已经成功启动

1.2 访问elasticsearch

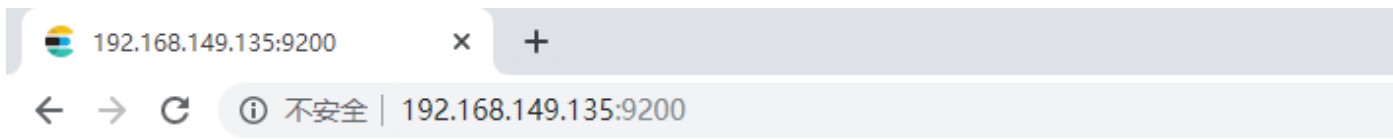
1、在访问elasticsearch前，请确保防火墙是关闭的，执行命令：

```
#暂时关闭防火墙
systemctl stop firewalld

# 或者

#永久设置防火墙状态
systemctl enable firewalld.service #打开防火墙永久性生效，重启后不会复原
systemctl disable firewalld.service #关闭防火墙，永久性生效，重启后不会复原
```

浏览器输入<http://192.168.149.135:9200/>，如下图



```
{
  "name" : "node-1",
  "cluster_name" : "my-application",
  "cluster_uuid" : "BXc2fK_NQVaiqsrcPLKhXg",
  "version" : {
    "number" : "7.4.0",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "22e1767283e61a198cb4db791ea66e3f11ab9910",
    "build_date" : "2019-09-27T08:36:48.569419Z",
    "build_snapshot" : false,
    "lucene_version" : "8.2.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

此时elasticsearch已成功启动：

重点几个关注下即可：

number" : "7.4.0" 表示elasticsearch版本
lucene_version" : "8.2.0" 表示lucene版本
name : 默认启动的时候指定了 ES 实例名称
cluster_name : 默认名为 elasticsearch

2 Elasticsearch辅助插件安装

2.1 Postman安装

1、什么是Postman

Postman是一个http模拟请求的工具。

官网介绍：“Modern software is built on APIs, Postman helps you develop APIs faster”

看得出来，它是一个专门测试 API 的工具，Postman 提供功能强大的 Web API 和 HTTP 请求的调试，它能够发送任何类型的HTTP 请求 (GET, POST, PUT, DELETE...), 并且能附带任何数量的参数和 Headers。不仅如此，它还提供测试数据和环境配置数据的导入导出。

进入官网www.getpostman.com，下载

2.2 Kibana安装

1、什么是Kibana

Kibana是一个针对Elasticsearch的开源分析及可视化平台，用来搜索、查看交互存储在Elasticsearch索引中的数据。使用Kibana，可以通过各种图表进行高级数据分析及展示。

Kibana让海量数据更容易理解。它操作简单，基于浏览器的用户界面可以快速创建仪表板（dashboard）实时显示Elasticsearch查询动态。

2、上传kibana

CRT中克隆一个窗口，上传Kibana

```
put E:\software\kibana-7.4.0-linux-x86_64.tar.gz
```

2、解压kibana

```
tar -xzf kibana-7.4.0-linux-x86_64.tar.gz -C /opt
```

解压到当前目录（/opt）下

3、修改kibana配置

```
vim /opt/kibana-7.4.0-linux-x86_64/config/kibana.yml
```

```
server.port: 5601
server.host: "0.0.0.0"
server.name: "kibana-itcast"
elasticsearch.hosts: ["http://127.0.0.1:9200"]
elasticsearch.requestTimeout: 99999
```

server.port: http访问端口

server.host: ip地址，0.0.0.0表示可远程访问

server.name: kibana服务名

elasticsearch.hosts: elasticsearch地址

elasticsearch.requestTimeout: 请求elasticsearch超时时间，默认为30000，此处可根据情况设置

4、启动kibana

由于kibana不建议使用root用户启动，如果用root启动，需要加--allow-root参数

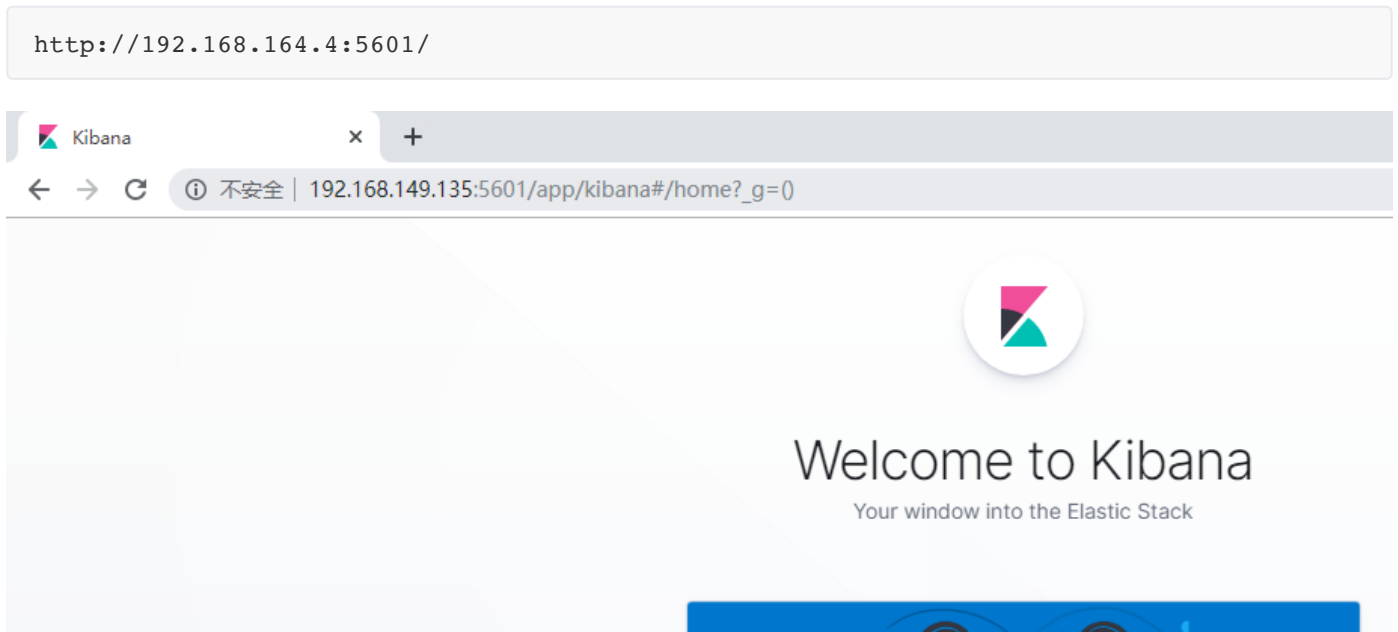
```
# 切换到kibana的bin目录
cd /opt/kibana-7.4.0-linux-x86_64/bin
# 启动
./kibana --allow-root
```

```
log [15:47:38.376] [info][migrations] Creating index .kibana_1.
log [15:47:59.482] [info][migrations] Creating index .kibana_task_manager_1.
log [15:48:04.027] [info][migrations] Pointing alias .kibana to .kibana_1.
log [15:48:04.089] [info][migrations] Pointing alias .kibana_task_manager to .kibana_task_manager_1.
log [15:48:04.338] [info][migrations] Finished in 27248ms.
log [15:48:04.341] [info][migrations] Finished in 4859ms.
log [15:48:04.354] [info][listening] Server running at http://0.0.0.0:5601
log [15:48:04.706] [info][server][Kibana][http] http server running at http://0.0.0.0:5601
log [15:48:06.502] [info][status][plugin:spaces@7.4.0] Status changed from yellow to green - Ready
```

启动成功。

5、访问kibana

1.浏览器输入<http://192.168.164.4:5601/>，如下图：



看到这个界面，说明Kibana已成功安装。

- Discover**：可视化查询分析器
- Visualize**：统计分析图表
- Dashboard**：自定义主面板（添加图表）
- Timelion**：Timelion是一个kibana时间序列展示组件（暂时不用）
- Dev Tools**：Console控制台（同CURL/POSTER，操作ES代码工具，代码提示，很方便）
- Management**：管理索引库(index)、已保存的搜索和可视化结果(save objects)、设置 kibana 服务器属性。

2.3 head安装

Tips:
课后扩展内容

head简介

head插件是ES的一个可视化管理插件，用来监视ES的状态，并通过head客户端和ES服务进行交互，比如创建映射、创建索引等。

在登陆和访问head插件地址和ElasticSearch前需要事先在服务器上安装和配置好ElasticSearch以及head插件。安装完后，默认head插件的web端口为9100，ElasticSearch服务的端口为9200，使用浏览器访问head地址，如<http://IP地址:9100/>，推荐使用Chrome浏览器，head插件对Chrome浏览器兼容更佳。进入head页面后将ElasticSearch连接输入框中填写正确的ElasticSearch服务地址，就可以监控ElasticSearch运行信息

2.3.1 Node安装

1) 什么是Node

简单的说 Node.js 就是运行在服务端的 JavaScript。Node.js 是一个基于 [Chrome V8](#) 引擎的 JavaScript 运行环境。Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型，使其轻量又高效。Node.js 的包管理器 [npm](#)，是全球最大的开源库生态系统。

2) 下载Node

上一节我们已经安装好了Elasticsearch，接下来我们来安装head插件，由于elasticsearch-head插件是由nodejs语言编写，所以安装elasticsearch-head前需要先安装nodejs。

首先，执行以下命令安装nodejs和grunt

打开虚拟机，执行wget命令下载Node，如下图：

```
wget https://nodejs.org/dist/v10.15.2/node-v10.15.2-linux-x64.tar.xz
```

```
[root@localhost opt]# wget https://nodejs.org/dist/v10.15.2/node-v10.15.2-linux-x64.tar.xz
--2019-10-16 01:27:04-- https://nodejs.org/dist/v10.15.2/node-v10.15.2-linux-x64.tar.xz
正在解析主机 nodejs.org (nodejs.org)... 104.20.22.46, 104.20.23.46, 2606:4700:10::6814:162e, ...
正在连接 nodejs.org (nodejs.org)[104.20.22.46]:443... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度: 12303736 (12M) [application/x-xz]
正在保存至: "node-v10.15.2-linux-x64.tar.xz"

100%[=====] 12,303,736 148KB/s 用时 70s

2019-10-16 01:28:16 (172 KB/s) - 已保存 "node-v10.15.2-linux-x64.tar.xz" [12303736/12303736]
```

3) 解压Node包

```
tar xvf node-v10.15.2-linux-x64.tar.xz
```

```
node-v10.15.2-linux-x64/include/node/openssl/conf.h
node-v10.15.2-linux-x64/include/node/openssl/md2.h
node-v10.15.2-linux-x64/include/node/openssl/pem2.h
node-v10.15.2-linux-x64/include/node/openssl/opensslv.h
node-v10.15.2-linux-x64/include/node/openssl/buffer.h
node-v10.15.2-linux-x64/include/node/openssl/ct.h
node-v10.15.2-linux-x64/include/node/openssl/hmac.h
node-v10.15.2-linux-x64/include/node/openssl/asn1t.h
node-v10.15.2-linux-x64/include/node/openssl/ui.h
node-v10.15.2-linux-x64/include/node/openssl/dh.h
node-v10.15.2-linux-x64/include/node/openssl/dtls1.h
node-v10.15.2-linux-x64/include/node/openssl/opensslconf_asm.h
node-v10.15.2-linux-x64/include/node/openssl/cms.h
node-v10.15.2-linux-x64/include/node/openssl/mdc2.h
node-v10.15.2-linux-x64/include/node/openssl/bio.h
node-v10.15.2-linux-x64/include/node/openssl/objects.h
node-v10.15.2-linux-x64/include/node/openssl/rc2.h
node-v10.15.2-linux-x64/include/node/openssl/ripemd.h
node-v10.15.2-linux-x64/include/node/v8config.h
node-v10.15.2-linux-x64/include/node/v8-inspector-protocol.h
node-v10.15.2-linux-x64/include/node/node_object_wrap.h
node-v10.15.2-linux-x64/include/node/v8-value-serializer-version.h
node-v10.15.2-linux-x64/include/node/v8-inspector.h
node-v10.15.2-linux-x64/include/node/node_buffer.h
node-v10.15.2-linux-x64/include/node/node.h
node-v10.15.2-linux-x64/include/node/v8-platform.h
node-v10.15.2-linux-x64/include/node/v8-util.h
node-v10.15.2-linux-x64/include/node/v8-version-string.h
node-v10.15.2-linux-x64/include/node/v8-profiler.h
node-v10.15.2-linux-x64/include/node/node_api.h
node-v10.15.2-linux-x64/include/node/node_version.h
node-v10.15.2-linux-x64/include/node/zlib.h
node-v10.15.2-linux-x64/include/node/zconf.h
[root@localhost opt]#
```

4) 设置软连接

解压文件的 bin 目录下包含了 node、npm 等命令，我们可以使用 ln 命令来设置软连接：

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

ln -s bin/node /usr/local/bin/
```

在/etc/profile中配置好path环境变量

```
vi ~/.bash_profile

export NODE_HOME=/opt/nodejs/node-v10.15.2-linux-x64

export PATH=$PATH:$NODE_HOME/bin
```

保存退出，使文件生效

```
source ~/.bash_profile
```

查看node安装版本，执行 node -v 验证安装如下图：

```
[root@localhost bin]# node -v
v8.16.2
```

2.3.2 grunt安装

安装grunt（运行在Node.js上面的任务管理器（task runner）），为了获得Grunt的更多产品特性，需要全局安装 Grunt's 命令行接口（CLI），使用npm进行安装，如下：

```
npm install -g grunt-cli
```

```
[root@localhost opt]# npm install -g grunt-cli
/usr/bin/grunt -> /usr/lib/node_modules/grunt-cli/bin/grunt
+ grunt-cli@1.3.2
updated 1 package in 23.395s
[root@localhost opt]#
```

查看grunt版本

```
[root@localhost opt]# grunt --version
grunt-cli v1.3.2
[root@localhost opt]#
```

输出grunt版本信息，表示安装成功。

2.3.3 head安装

1) 执行命令安装git

```
git yum install git -y
```



```

验证中      : 1:perl-Error-0.17020-2.el7.noarch      18/32
验证中      : git-1.8.3.1-20.el7.x86_64            19/32
验证中      : perl-Storable-2.45-3.el7.x86_64       20/32
验证中      : perl-Scalar-List-Utils-1.27-248.el7.x86_64 21/32
验证中      : perl-Pod-Usage-1.63-3.el7.noarch     22/32
验证中      : perl-Encode-2.51-7.el7.x86_64       23/32
验证中      : perl-Pod-Perldoc-3.20-4.el7.noarch   24/32
验证中      : perl-podlators-2.5.1-3.el7.noarch    25/32
验证中      : perl-File-Path-2.09-2.el7.noarch     26/32
验证中      : perl-threads-1.87-4.el7.x86_64      27/32
验证中      : 1:perl-Pod-Simple-3.28-4.el7.noarch  28/32
验证中      : perl-Filter-1.49-3.el7.x86_64       29/32
验证中      : perl-Getopt-Long-2.40-3.el7.noarch   30/32
验证中      : perl-Text-ParseWords-3.29-4.el7.noarch 31/32
验证中      : 4:perl-5.16.3-294.el7_6.x86_64      32/32

已安装:
git.x86_64 0:1.8.3.1-20.el7

作为依赖被安装:
perl.x86_64 4:5.16.3-294.el7_6      perl-Carp.noarch 0:1.26-244.el7      perl-Encode.x86_64 0:2.51-7.el7
perl-Error.noarch 1:0.17020-2.el7    perl-Exporter.noarch 0:5.68-3.el7    perl-File-Path.noarch 0:2.09-2.el7
perl-File-Temp.noarch 0:0.23.01-3.el7 perl-Filter.x86_64 0:1.49-3.el7      perl-Getopt-Long.noarch 0:2.40-3.el7
perl-Git.noarch 0:1.8.3.1-20.el7     perl-HTTP-Tiny.noarch 0:0.033-3.el7  perl-PathTools.x86_64 0:3.40-5.el7
perl-Pod-Escapes.noarch 1:1.04-294.el7_6 perl-Pod-Perldoc.noarch 0:3.20-4.el7  perl-Pod-Simple.noarch 1:3.28-4.el7
perl-Pod-Usage.noarch 0:1.63-3.el7    perl-Scalar-List-Utils.x86_64 0:1.27-248.el7 perl-Socket.x86_64 0:2.010-4.el7
perl-Storable.x86_64 0:2.45-3.el7     perl-TermReadKey.x86_64 0:2.30-20.el7 perl-Text-ParseWords.noarch 0:3.29-4.el7
perl-Time-HiRes.x86_64 4:1.9725-3.el7 perl-Time-Local.noarch 0:1.2300-2.el7 perl-constant.noarch 0:1.27-2.el7
perl-libs.x86_64 4:5.16.3-294.el7_6  perl-macros.x86_64 4:5.16.3-294.el7_6 perl-parent.noarch 1:0.225-244.el7
perl-podlators.noarch 0:2.5.1-3.el7   perl-threads.x86_64 0:1.87-4.el7     perl-threads-shared.x86_64 0:1.43-6.el7
rsync.x86_64 0:3.1.2-6.el7_6.1

完毕!

```

2) 切换到/opt目录下,执行下面的克隆命令

```
git clone git://github.com/mobz/elasticsearch-head.git
```

```

[root@localhost opt]# git clone git://github.com/mobz/elasticsearch-head.git
正克隆到 'elasticsearch-head'...
remote: Enumerating objects: 77, done.
remote: Counting objects: 100% (77/77), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 4337 (delta 38), reused 45 (delta 17), pack-reused 4260
接收对象中: 100% (4337/4337), 2.51 MiB | 111.00 KiB/s, done.
处理 delta 中: 100% (2411/2411), done.
[root@localhost opt]#

```

3) 进入到elasticsearch-head目录

```
cd elasticsearch-head
```

4) 运行

在运行之前我们需要修改下elasticsearch.yml, 因为ES默认不开启跨域访问, 需要添加以下配置:

```

#开启cors跨域访问支持, 默认为false
http.cors.enabled: true
#跨域访问允许的域名地址, (允许所有域名)以上使用正则
http.cors.allow-origin: "*"

```

然后开始执行运行命令:

```
npm run start
```

```
[root@localhost elasticsearch-head]# npm run start

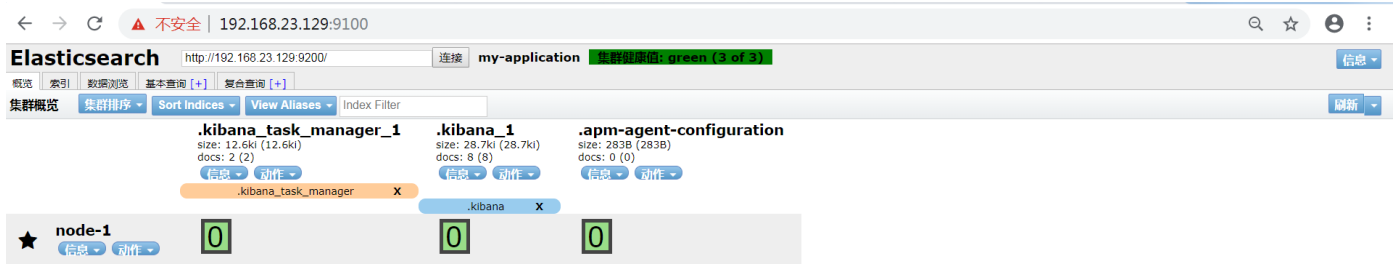
> elasticsearch-head@0.0.0 start /opt/elasticsearch-head
> grunt server

>> Local Npm module "grunt-contrib-jasmine" not found. Is it installed?

Running "connect:server" (connect) task
Waiting forever...
Started connect web server on http://localhost:9100
```

5) 访问head

浏览器输入ip:port:9100，如下图



看到这个界面说明我们的head插件成功安装并且成功连接Elasticsearch。