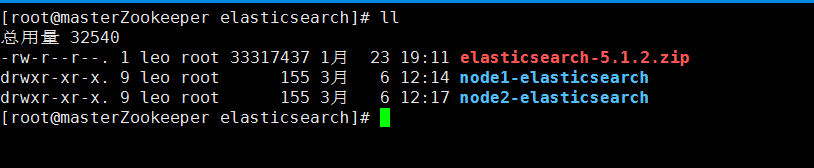
# ElasticSearch

下载elasticsearch

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

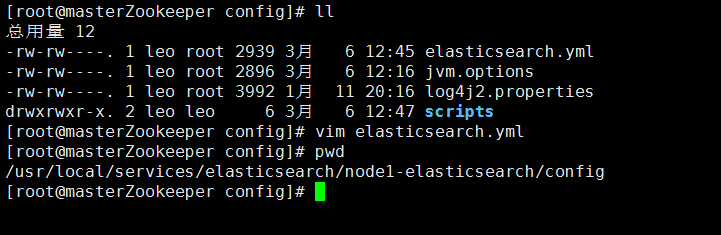
放到安装目录 /usr/local/services/elasticsearch

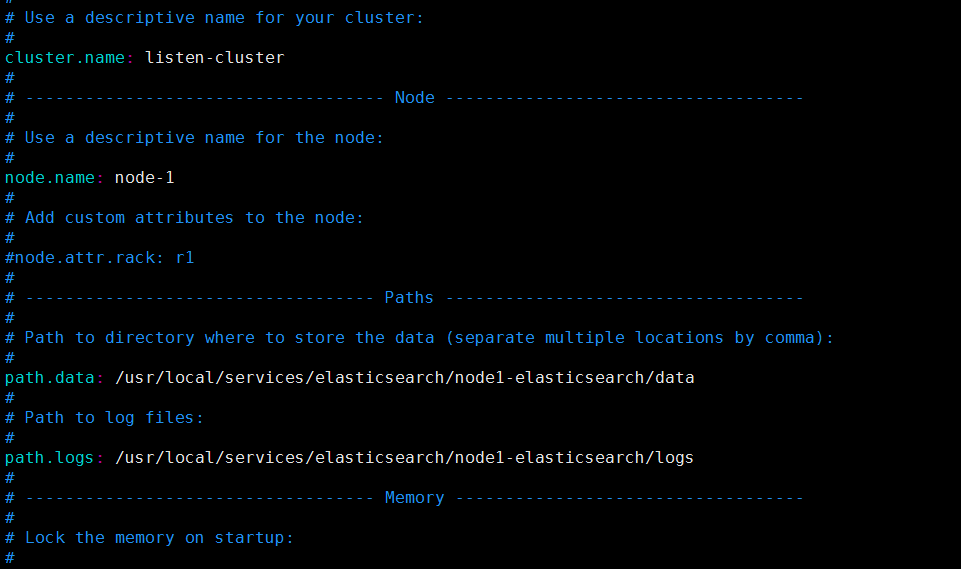
解压 复制出两个节点

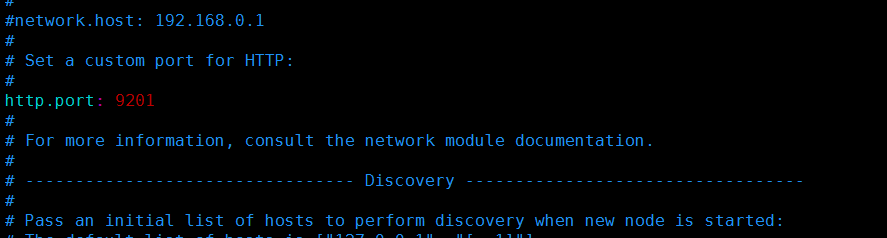


修改配置文件

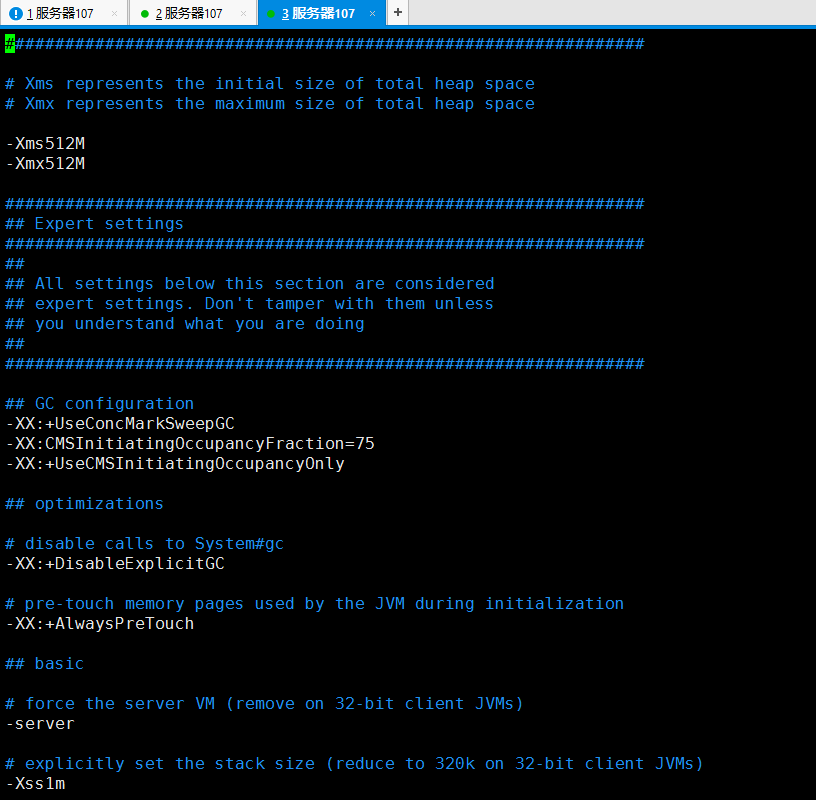
## vim elasticsearch.yml



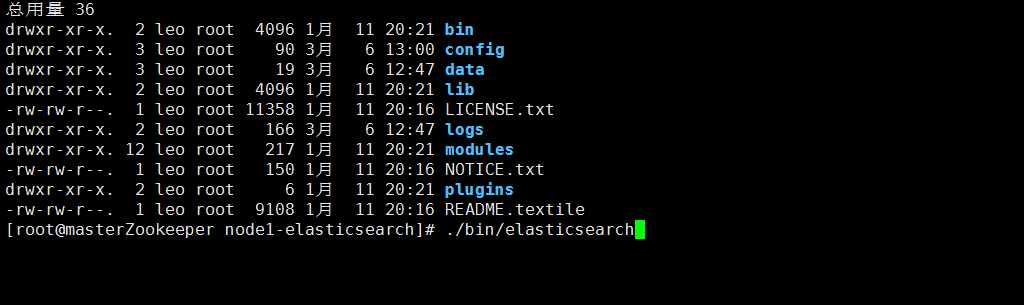




## vim jvm.options

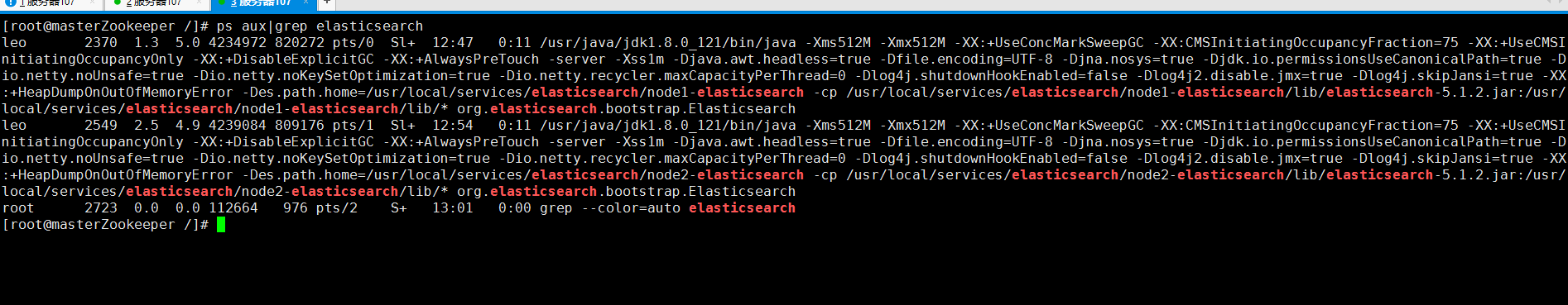


启动elasticsearch



查看进程

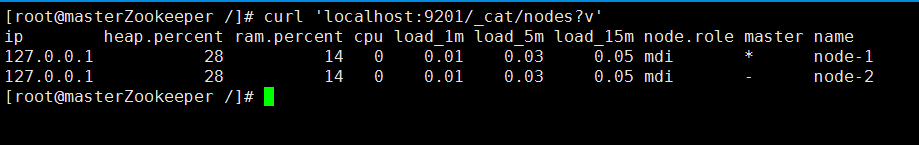
## ps aux|grep elasticsearch



## 简单测试:

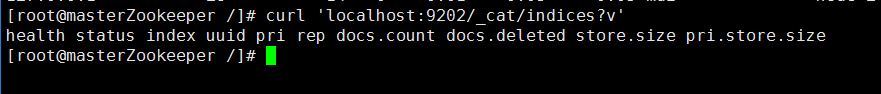
#查找所有节点

curl 'localhost:9201/\_cat/nodes?v'



#查找所有索引库

curl 'localhost:9202/\_cat/indices?v'



#创建新的索引

curl -XPUT 'localhost:9200/customer?pretty'

#查看创建的索引库

curl 'localhost:9201/\_cat/indices?v'

#往索引库中添加一个文档

curl -XPUT 'localhost:9205/customer/external/1?pretty' -d '

　　{

　　 "name": "John Doe"

　　}'

#查看添加的文档

curl -XGET 'localhost:9201/customer/external/1?pretty'

#删除索引库

curl -XDELETE 'localhost:9201/customer?pretty'

#查看是否已经删除

curl 'localhost:9201/\_cat/indices?v'

# 基础概念

    Elasticsearch有几个核心概念。从一开始理解这些概念会对整个学习过程有莫大的帮助。

## 接近实时（NRT）

        Elasticsearch是一个接近实时的搜索平台。这意味着，从索引一个文档直到这个文档能够被搜索到有一个轻微的延迟（通常是1秒）。

## 集群（cluster）

        一个集群就是由一个或多个节点组织在一起，它们共同持有你整个的数据，并一起提供索引和搜索功能。一个集群由一个唯一的名字标识，这个名字默认就是“elasticsearch”。这个名字是重要的，因为一个节点只能通过指定某个集群的名字，来加入这个集群。在产品环境中显式地设定这个名字是一个好习惯，但是使用默认值来进行测试/开发也是不错的。

## 节点（node）

        一个节点是你集群中的一个服务器，作为集群的一部分，它存储你的数据，参与集群的索引和搜索功能。和集群类似，一个节点也是由一个名字来标识的，默认情况下，这个名字是一个随机的漫威漫画角色的名字，这个名字会在启动的时候赋予节点。这个名字对于管理工作来说挺重要的，因为在这个管理过程中，你会去确定网络中的哪些服务器对应于Elasticsearch集群中的哪些节点。  
          
        一个节点可以通过配置集群名称的方式来加入一个指定的集群。默认情况下，每个节点都会被安排加入到一个叫做“elasticsearch”的集群中，这意味着，如果你在你的网络中启动了若干个节点，并假定它们能够相互发现彼此，它们将会自动地形成并加入到一个叫做“elasticsearch”的集群中。  
          
        在一个集群里，只要你想，可以拥有任意多个节点。而且，如果当前你的网络中没有运行任何Elasticsearch节点，这时启动一个节点，会默认创建并加入一个叫做“elasticsearch”的集群。

## 索引（index）

          
        一个索引就是一个拥有几分相似特征的文档的集合。比如说，你可以有一个客户数据的索引，另一个产品目录的索引，还有一个订单数据的索引。一个索引由一个名字来标识（必须全部是小写字母的），并且当我们要对对应于这个索引中的文档进行索引、搜索、更新和删除的时候，都要使用到这个名字。  
          
        在一个集群中，如果你想，可以定义任意多的索引。

## 类型（type）

        在一个索引中，你可以定义一种或多种类型。一个类型是你的索引的一个逻辑上的分类/分区，其语义完全由你来定。通常，会为具有一组共同字段的文档定义一个类型。比如说，我们假设你运营一个博客平台并且将你所有的数据存储到一个索引中。在这个索引中，你可以为用户数据定义一个类型，为博客数据定义另一个类型，当然，也可以为评论数据定义另一个类型。

## 文档（document）

          
        一个文档是一个可被索引的基础信息单元。比如，你可以拥有某一个客户的文档，某一个产品的一个文档，当然，也可以拥有某个订单的一个文档。文档以JSON（Javascript Object Notation）格式来表示，而JSON是一个到处存在的互联网数据交互格式。  
          
        在一个index/type里面，只要你想，你可以存储任意多的文档。注意，尽管一个文档，物理上存在于一个索引之中，文档必须被索引/赋予一个索引的type。

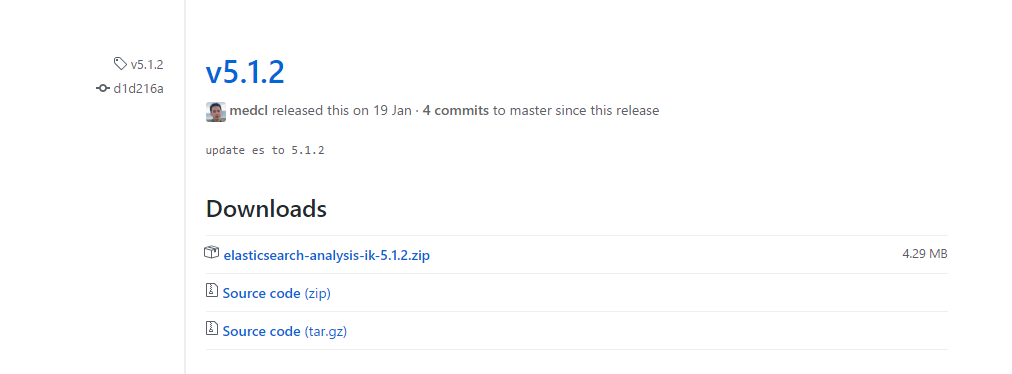
## 分片和复制（shards & replicas）

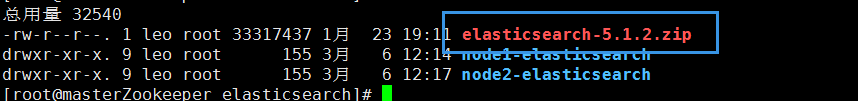
          
        一个索引可以存储超出单个结点硬件限制的大量数据。比如，一个具有10亿文档的索引占据1TB的磁盘空间，而任一节点都没有这样大的磁盘空间；或者单个节点处理搜索请求，响应太慢。  
          
        为了解决这个问题，Elasticsearch提供了将索引划分成多份的能力，这些份就叫做分片。当你创建一个索引的时候，你可以指定你想要的分片的数量。每个分片本身也是一个功能完善并且独立的“索引”，这个“索引”可以被放置到集群中的任何节点上。  
          
        分片之所以重要，主要有两方面的原因：  
          
            - 允许你水平分割/扩展你的内容容量  
            - 允许你在分片（潜在地，位于多个节点上）之上进行分布式的、并行的操作，进而提高性能/吞吐量  
          
        至于一个分片怎样分布，它的文档怎样聚合回搜索请求，是完全由Elasticsearch管理的，对于作为用户的你来说，这些都是透明的。  
          
        在一个网络/云的环境里，失败随时都可能发生，在某个分片/节点不知怎么的就处于离线状态，或者由于任何原因消失了，这种情况下，有一个故障转移机制是非常有用并且是强烈推荐的。为此目的，Elasticsearch允许你创建分片的一份或多份拷贝，这些拷贝叫做复制分片，或者直接叫复制。  
          
        复制之所以重要，有两个主要原因：  
            - 在分片/节点失败的情况下，提供了高可用性。因为这个原因，注意到复制分片从不与原/主要（original/primary）分片置于同一节点上是非常重要的。  
            - 扩展你的搜索量/吞吐量，因为搜索可以在所有的复制上并行运行  
              
        总之，每个索引可以被分成多个分片。一个索引也可以被复制0次（意思是没有复制）或多次。一旦复制了，每个索引就有了主分片（作为复制源的原来的分片）和复制分片（主分片的拷贝）之别。分片和复制的数量可以在索引创建的时候指定。在索引创建之后，你可以在任何时候动态地改变复制的数量，但是你事后不能改变分片的数量。  
          
        默认情况下，Elasticsearch中的每个索引被分片5个主分片和1个复制，这意味着，如果你的集群中至少有两个节点，你的索引将会有5个主分片和另外5个复制分片（1个完全拷贝），这样的话每个索引总共就有10个分片。  
          
        这些问题搞清楚之后，我们就要进入好玩的部分了...

# 分词器ik安装

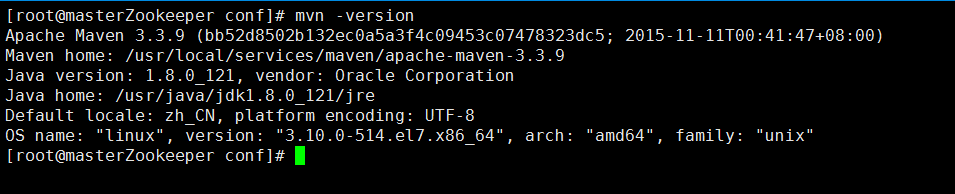
要下载自己所使用es所对应的版本

<https://github.com/medcl/elasticsearch-analysis-ik/releases>



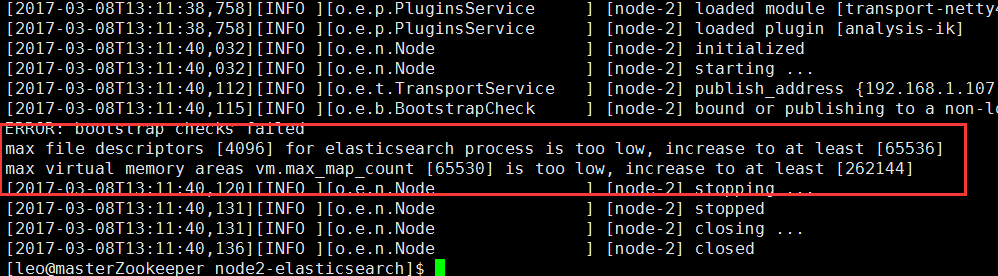


安装好maven



## 遇到问题

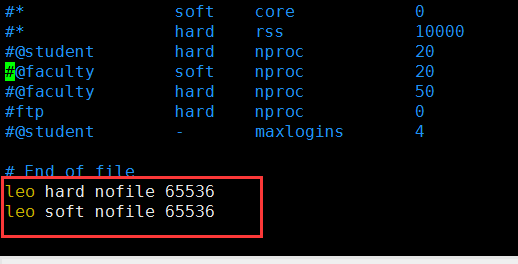
max file descriptors [4096] for elasticsearch process is too low, increase to at least [65536]



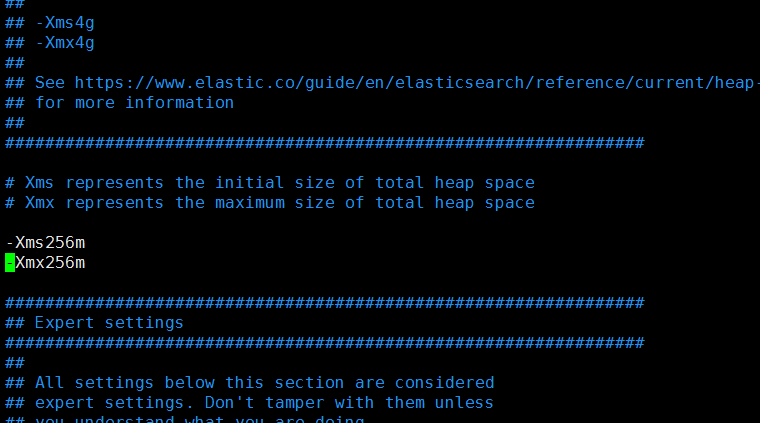
## 解决问题:

vim elasticsearch.yml

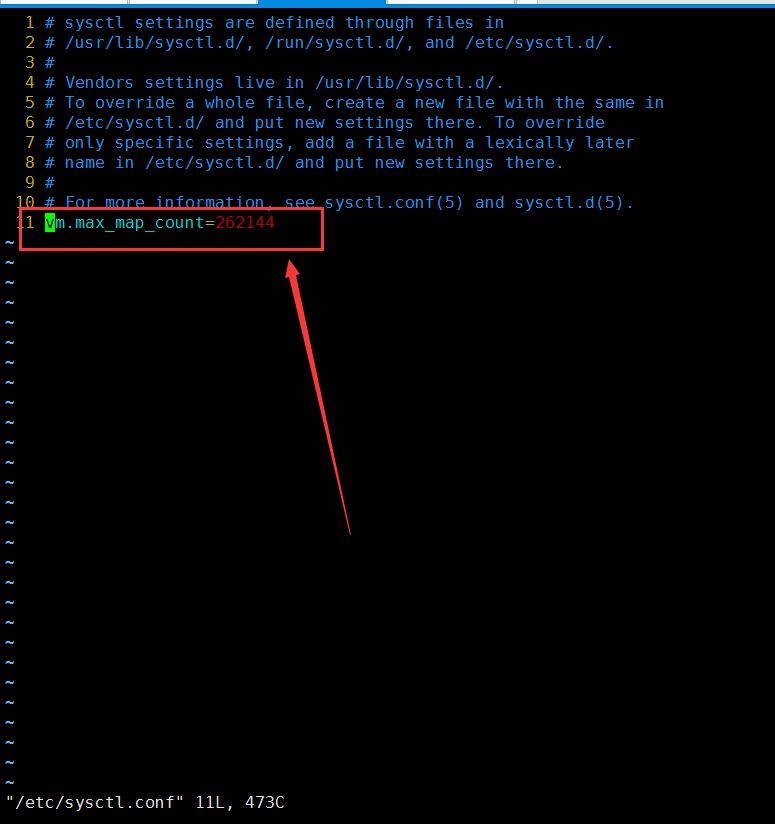
vim /etc/security/limits.conf



vim jvm.options



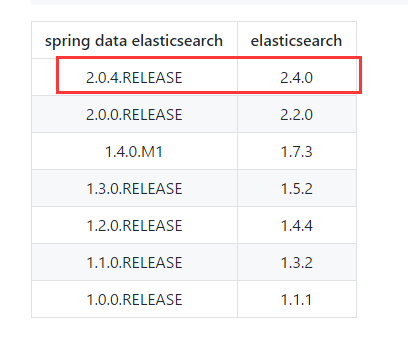
vim /etc/sysctl.conf



# 版本问题

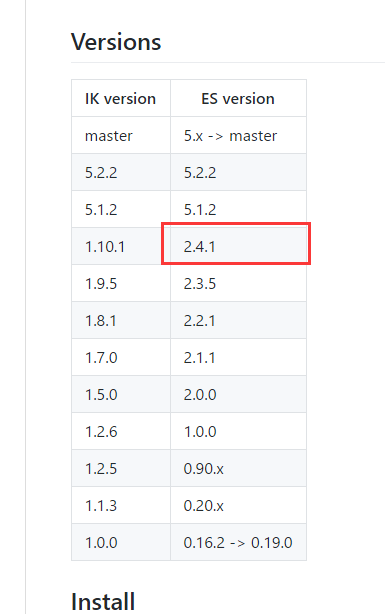
Spring-data-elasticsearch目前能支持最高版本的elasticsearch是2.4.0

<https://github.com/merlinxqh/spring-data-elasticsearch>



Ik分词器 又找不到与之对应的版本这就尴尬了

<https://github.com/merlinxqh/elasticsearch-analysis-ik>



## 解决办法

网上找了个方法 从

<https://github.com/merlinxqh/elasticsearch-analysis-ik>

下载与 elasticsearch-2.4.0相近的 版本

[elasticsearch-analysis-ik](https://github.com/merlinxqh/elasticsearch-analysis-ik)**-1.10.1 然后将其pom.xml中elasticsearch版本号改为2.4.0**

mvn clean package

打包后 到 target/releases 目录 有[elasticsearch-analysis-ik](https://github.com/merlinxqh/elasticsearch-analysis-ik)**-1.10.1.zip文件**

**放至es安装目录/plugins/ik (没有就创建) 解压即可**

**可以通过 ./bin/plugin list 命令查看**

# Elasticsearch配置详解

摘自:

<http://www.cnblogs.com/wangxue2/p/5773799.html>

## ****一、Cluster  setting****

### Cluster

**indices.ttl.interval**  允许设置多久过期的文件会被自动删除。默认值是60秒。

**indices.cache.filter.size**  ES的filter cache有两种，一种是node级别的cache（filter cache默认类型），一种是index级别的filter cache。Node级别的cache被整个node共享，并且可以使用百分比设置，对应的属性为index.cache.filter.size，这个属性的值可以使百分比，也可以是具体的大小。Index级别的cache，顾名思义，就是针对单个索引的大小。ES官方并不推荐使用这种设置，因为谁也无法预测索引级别的缓存到底有多大（可能非常大，超过了node的内存），一个索引可能分布在多个node上面，而多个node的结果如果汇总到一个node上，其结果可想而知。默认值是10%。

**discovery.zen.minimum\_master\_nodes**        避免脑裂现象（由于某些节点的失效，部分节点的网络连接会断开，并形成一个与原集群一样的集群，这种情况称为集群脑裂现象）discovery.zen.minimum\_master\_nodes参数决定了要选举一个master需要多少个节点（最少候选节点数）。默认值是1。根据一般经验这个一般设置成N/2+1（向下取整），N是集群中节点的数量。

### routing

**node\_initial\_primaries\_recoveries**每个初选节点允许控制初始修复的具体数量

**cluster\_concurrent\_rebalance**控制集群宽度允许的分片平衡可以并发多少

**awareness.attributes**集群配置意识允许配置分片和副本分配在与节点相关联的通用属性

**node\_concurrent\_recoveries**每个节点上允许多少个并发修复，默认是2

**disable\_allocation**允许禁用主要分配

**disable\_replica\_allocation** 允许禁用副本分配

### recovery

**concurrent\_streams**设置当从对等中恢复碎片时限制打开的并发流的数量 默认5

**file\_chunk\_size**文件块大小  默认512kb

**translog\_ops**

**translog\_size**

**max\_bytes\_per\_sec**设置恢复时每分钟油门的吞吐量  默认20mb

**compress**启用压缩为所有节点间的通信   默认禁用

## **二、Elasticsearch Configuration Example**

#####################Elasticsearch Configuration Example #####################

# This file contains an overview of various configuration settings,

#本文件针对操作人员包含了对不同配置设置的概述

# targeted at operations staff. Application developers should

# consult the guide at <http://elasticsearch.org/guide>.

#应用程序开发人员应该参考<http://elasticsearch.org/guide>指南

# The installation procedure is covered at

# <http://elasticsearch.org/guide/en/elasticsearch/reference/current/setup.html>.

#

#安装过程在<http://elasticsearch.org/guide/en/elasticsearch/reference/current/setup.html>

# Elasticsearch comes with reasonable defaults for most settings,

# so you can try it out without bothering with configuration.

#

#ela本身附带最合理的默认值设置，所以你可以不用修改配置文件直接使用

# Most of the time, these defaults are just fine for running a production

# cluster. If you're fine-tuning your cluster, or wondering about the

#  effect of certain configuration option, please \_do ask\_ on the

# mailing list or IRC channel [http://elasticsearch.org/community].

#大多时候，这些默认设置可以很好的运行生产环境集群。

#如果你微调你的集群或者考虑到某些配置选项的效果，请在邮件列表或者IRC频道[http://elasticsearch.org/community]询问。

# Any element in the configuration can be replaced with environment variables

# by placing them in ${...} notation. For example:

#

#任何组成的配置可以使用环境变量${…}替换他们，例如：

#node.rack: ${RACK\_ENV\_VAR}

# For information on supported formats and syntax for the config file, see

#<http://elasticsearch.org/guide/en/elasticsearch/reference/current/setup-configuration.html>

#支持的格式和语法信息的配置文件，请参考<http://elasticsearch.org/guide/en/elasticsearch/reference/current/setup-configuration.html>

## **三、Cluster**

################################### Cluster ###################################

################################## 集群设置###################################

# Cluster name identifies your cluster for auto-discovery. If you're running

# multiple clusters on the same network, make sure you're using unique names.

#

#集群名称识别集群自动发现。如果在同一网段运行多个集群，要确保他们使用唯一的名称。

cluster.name: log-center-it-test

#集群名称

cluster.routing.allocation.disk.watermark.low: "90%"

cluster.routing.allocation.disk.watermark.high: "96%"

indices.fielddata.cache.size: "30%"

索引字段数据缓存大小

## **四、Node**

#################################### Node #####################################

################################## 节点设置###################################

# Node names are generated dynamically on startup, so you're relieved

# from configuring them manually. You can tie this node to a specific name:

#

#节点名称是在启动的时候动态生成的，免去了手动配置。你也可以为这个节点取一个特定的名字:

node.name: "ip"

# Every node can be configured to allow or deny being eligible as the master,

# and to allow or deny to store the data.

#

#每个节点都可以被配置为允许或者拒绝成为主节点，并且允许或拒绝存储数据。

# Allow this node to be eligible as a master node (enabled by default):

#

#允许这个节点成为主节点（默认是允许的）

#node.master: true

#

# Allow this node to store data (enabled by default):

#

#允许这个节点存储数据（默认是允许的）

#node.data: true

# You can exploit these settings to design advanced cluster topologies.

#

#你可以利用这些设置设计先进的集群拓扑。

# 1. You want this node to never become a master node, only to hold data.

#    This will be the "workhorse" of your cluster.

#

#1.不想这个节点成为主节点，只作为存储数据节点。它会成为集群的驮马（重负荷机器）

node.master: false

node.data: true

#

# 2. You want this node to only serve as a master: to not store any data and

#    to have free resources. This will be the "coordinator" of your cluster.

#

#2.想要这个节点只作为主节点，不存储任何数据和有免费资源。他会成为集群的协调员

#node.master: true

#node.data: false

#

# 3. You want this node to be neither master nor data node, but

#    to act as a "search load balancer" (fetching data from nodes,

#    aggregating results, etc.)

#

#3.想要这个节点既不作为主节点也不作为数据节点，只作为搜索负载平衡器（从节点获取数据，聚合结果，等等）

#node.master: false

#node.data: false

# Use the Cluster Health API [http://localhost:9200/\_cluster/health], the

# Node Info API [http://localhost:9200/\_nodes] or GUI tools

# such as <http://www.elasticsearch.org/overview/marvel/>,

# <http://github.com/karmi/elasticsearch-paramedic>,

# <http://github.com/lukas-vlcek/bigdesk> and

# <http://mobz.github.com/elasticsearch-head> to inspect the cluster state.

#使用集群健康API，节点信息API或者GUI工具，如。。、。。和。。检查集群的状态。

# A node can have generic attributes associated with it, which can later be used

# for customized shard allocation filtering, or allocation awareness. An attribute

# is a simple key value pair, similar to node.key: value, here is an example:

#

#一个节点可以有与后来被用于定制的碎片分配过滤或分配意识相关联的通用的属性，一个属性是一个简单的键值对，类似于节点。

#node.rack: rack314

# By default, multiple nodes are allowed to start from the same installation location

# to disable it, set the following:

#默认的，多个节点可以从相同的安装位置来禁用它，设置如下:

#node.max\_local\_storage\_nodes: 1

## **五、Index**

#################################### Index ####################################

# You can set a number of options (such as shard/replica options, mapping

# or analyzer definitions, translog settings, ...) for indices globally,

# in this file.

#

#可以在这个文件中为全局索引设置一些选项（如分片/副本操作，映像或分析仪定义，translog设置）

# Note, that it makes more sense to configure index settings specifically for

# a certain index, either when creating it or by using the index templates API.

#

#注意，当创建或者通过使用API索引模板，为某一个特定的索引配置索引会是更有意义的配置。

#See<http://elasticsearch.org/guide/en/elasticsearch/reference/current/index-modules.html> and

#<http://elasticsearch.org/guide/en/elasticsearch/reference/current/indices-create-index.html>

# for more information.

# Set the number of shards (splits) of an index (5 by default):

#

#设置一个索引的分片数（默认是5）：

#index.number\_of\_shards: 5

# Set the number of replicas (additional copies) of an index (1 by default):

#

#设置一个索引的副本数（默认是1）：

#index.number\_of\_replicas: 1

# Note, that for development on a local machine, with small indices, it usually

# makes sense to "disable" the distributed features:

#

#注意，在本地机器上开发，一个小的索引通常会有意义的禁用分布式的特性（通俗易懂的说就是不需要副本了，因为索引太小，没必要建副本。）

#index.number\_of\_shards: 1

#index.number\_of\_replicas: 0

# These settings directly affect the performance of index and search operations

# in your cluster. Assuming you have enough machines to hold shards and

# replicas, the rule of thumb is replicas, the rule of thumb is:

#

#这些设置直接影响集群中索引和搜索操作的性能。假如你有足够机器来存储分片和副本，拇指律（作为一项经验法则）是副本，拇指律是：

# 1. Having more \*shards\* enhances the \_indexing\_ performance and allows to

#    \_distribute\_ a big index across machines.

# 2. Having more \*replicas\* enhances the \_search\_ performance and improves the

#    cluster \_availability\_.

#

#1.有很多的分片提高索引的性能并且允许在机器上分配更大的索引。

#2.有更多的副本提高搜索的性能并且提高集群的可利用性。

# The "number\_of\_shards" is a one-time setting for an index.

#

# "number\_of\_shards"对一个索引的一次性设置

# The "number\_of\_replicas" can be increased or decreased anytime,

# by using the Index Update Settings API.

#

#"number\_of\_replicas"可以在任何时间增加或者取消，通过使用更新设置索引API

# Elasticsearch takes care about load balancing, relocating, gathering the

# results from nodes, etc. Experiment with different settings to fine-tune

# your setup.

#ela关注的是负载均衡，迁移，来自节点收集结果，等。尝试不同的设置来调整你的设置。

# Use the Index Status API (<http://localhost:9200/A/\_status>) to inspect

# the index status.

#用索引状态API来检测索引状态。

## **六、Paths**

################################### Paths ####################################

################################# 路径设置##################################

# Path to directory containing configuration (this file and logging.yml):

#

#配置文件的路径

#path.conf: /path/to/conf

# Path to directory where to store index data allocated for this node.

#

#存储为该节点分配的索引数据的路径

#path.data: /path/to/data

#

# Can optionally include more than one location, causing data to be striped across

# the locations (a la RAID 0) on a file level, favouring locations with most free

# space on creation. For example:

#

#能够包含多个位置，导致数据在整个范围在文件级被剥离，支持在创新的时候可以有最自由的空间（并不懂什么意思）

#path.data: /path/to/data1,/path/to/data2

# Path to temporary files:

#

#临时文件的路径

#path.work: /path/to/work

# Path to log files:

#

#日志文件的路径

#path.logs: /path/to/logs

# Path to where plugins are installed:

#

#安装插件的路径

#path.plugins: /path/to/plugins

## **七、Plugin**

#################################### Plugin ###################################

#################################### 插件 ###################################

# If a plugin listed here is not installed for current node, the node will not start.

#

#如果这里列出的插件没有安装在当前节点，该节点将无法启动

#plugin.mandatory: mapper-attachments,lang-groovy

## **八、Memory**

################################### Memory ####################################

################################### 内存 ####################################

# Elasticsearch performs poorly when JVM starts swapping: you should ensure that

# it \_never\_ swaps.

#

#当JVM开始启动时，ela执行很差：应该确保它是it \_never\_ swaps

# Set this property to true to lock the memory:

#

#将这个属性设置为true来锁定内存

#bootstrap.mlockall: true

bootstrap.mlockall: true

# Make sure that the ES\_MIN\_MEM and ES\_MAX\_MEM environment variables are set

# to the same value, and that the machine has enough memory to allocate

#  for Elasticsearch, leaving enough memory for the operating system itself.

#

#确保ES\_MIN\_MEM和ES\_MAX\_MEM环境变量设置为相同的值，而机器为ela有足够的内存分配，为操作系统本身留下足够的内存。

# You should also make sure that the Elasticsearch process is allowed to lock

# the memory, eg. by using `ulimit -l unlimited`.

#应该确保ela过程允许锁定内存，如通过使用`ulimit -l unlimited`

## **九、Network And HTTP**

############################## Network And HTTP ###############################

##############################网络和HTTP设置 ###############################

# Elasticsearch, by default, binds itself to the 0.0.0.0 address, and listens

# on port [9200-9300] for HTTP traffic and on port [9300-9400] for node-to-node

# communication. (the range means that if the port is busy, it will automatically

# try the next port).

#ela默认情况下，结合本身的0.0.0.0地址和端口上侦听http传输（9200-9300）端口和（9300-9400）为节点到节点通信传输端口。（这意味着如果端口繁忙，他会自动尝试下一个端口）。

# Set the bind address specifically (IPv4 or IPv6):

#

#设置绑定地址（IPv4或者IPv6）：

#network.bind\_host: 192.168.0.1

# Set the address other nodes will use to communicate with this node. If not

# set, it is automatically derived. It must point to an actual IP address.

#

#设置地址其他节点将使用此地址与这个节点通信，如果没有设置，他会自动派生。必须指向一个实际的IP地址。

#network.publish\_host: 192.168.0.1

# Set both 'bind\_host' and 'publish\_host':

#

#设置绑定主机和发布主机：

#network.host: 192.168.0.1

# Set a custom port for the node to node communication (9300 by default):

#

#为节点到节点通信设置自定义端口（默认是9300）

#transport.tcp.port: 9300

# Enable compression for all communication between nodes (disabled by default):

#

#启用压缩为所有节点之间通信（默认情况下禁用）：

#transport.tcp.compress: true

# Set a custom port to listen for HTTP traffic:

#

#设置HTTP传输端口

#http.port: 9200

# Set a custom allowed content length:

#

#设置一个自定义允许内容长度

#http.max\_content\_length: 100mb

# Disable HTTP completely:

#

#禁用HTTP完全

#http.enabled: false

## **十、Gateway**

################################### Gateway ###################################

# The gateway allows for persisting the cluster state between full cluster

# restarts. Every change to the state (such as adding an index) will be stored

# in the gateway, and when the cluster starts up for the first time,

# it will read its state from the gateway.

#网关允许坚持全面重启集群之间的集群状态。每一个变化的状态（例如添加索引）将存储在网关中，当集群首次启动时，它将从网关读取它的状态。

# There are several types of gateway implementations. For more information, see

# <http://elasticsearch.org/guide/en/elasticsearch/reference/current/modules-gateway.html>.

# The default gateway type is the "local" gateway (recommended):

#

#有几种类型的网关的实现。更多的信息请参见。。

#默认的网关类型是本地网关（推荐）：

#gateway.type: local

# Settings below control how and when to start the initial recovery process on

# a full cluster restart (to reuse as much local data as possible when using shared

# gateway).

#设置在集群重启时控制如何什么时间开始最初的恢复进程（当使用分享网关时尽可能的重用本地数据）

# Allow recovery process after N nodes in a cluster are up:

#

#集群中的N个节点启动之后允许恢复进程：

#gateway.recover\_after\_nodes: 1

# Set the timeout to initiate the recovery process, once the N nodes

# from previous setting are up (accepts time value):

#

#为启动恢复过程设置超时，一旦先前设置的n个节点开启（接受时间价值）：

#gateway.recover\_after\_time: 5m

# Set how many nodes are expected in this cluster. Once these N nodes

# are up (and recover\_after\_nodes is met), begin recovery process immediately

# (without waiting for recover\_after\_time to expire):

#

#设置集群中期望有多少个结节点，一旦这些节点启动，立即开始恢复过程。

#gateway.expected\_nodes: 2

**十一、Recovery Throttling**

############################# Recovery Throttling #############################

############################# 恢复限流 #############################

# These settings allow to control the process of shards allocation between

# nodes during initial recovery, replica allocation, rebalancing,

# or when adding and removing nodes.

#这些设置允许在最初恢复、副本分配、平衡或者添加删除节点时在节点之间控制分片的过程，

# Set the number of concurrent recoveries happening on a node:

#

#设置一个节点上并发复苏的数量

# 1. During the initial recovery

#

#1.最初恢复

#cluster.routing.allocation.node\_initial\_primaries\_recoveries: 4

#

# 2. During adding/removing nodes, rebalancing, etc

#

#2.在添加、删除节点，再平衡，等等

#cluster.routing.allocation.node\_concurrent\_recoveries: 2

# Set to throttle throughput when recovering (eg. 100mb, by default 20mb):

#

#当恢复的时候设置油门吞吐量（例如100mb，默认20mb）

#indices.recovery.max\_bytes\_per\_sec: 20mb

# Set to limit the number of open concurrent streams when

# recovering a shard from a peer:

#

#当分片从同位体中恢复时设置限制打开并发流的数量

#indices.recovery.concurrent\_streams: 5

## **十二、Discovery**

################################## Discovery ##################################

################################## 发现 ##################################

# Discovery infrastructure ensures nodes can be found within a cluster

# and master node is elected. Multicast discovery is the default.

#发现基础设施确保节点能够在集群中被发现并且主节点被选出。多播发现是默认的。

# Set to ensure a node sees N other master eligible nodes to be considered

# operational within the cluster. This should be set to a quorum/majority of

# the master-eligible nodes in the cluster.

#

#确保一个节点在集群中和在N个其他有资格的节点中被认为是操作者。#discovery.zen.minimum\_master\_nodes: 1

# Set the time to wait for ping responses from other nodes when discovering.

# Set this option to a higher value on a slow or congested network

# to minimize discovery failures:

#

#当发现节点的时候设置等待来自其他节点的ping的回应的时间

#设置这个操作在缓慢和拥挤的网络中达到一个更高的值：

#discovery.zen.ping.timeout: 3s

discovery.zen.ping.timeout: 120s

# For more information, see

# <http://elasticsearch.org/guide/en/elasticsearch/reference/current/modules-discovery-zen.html>

# Unicast discovery allows to explicitly control which nodes will be used

# to discover the cluster. It can be used when multicast is not present,

# or to restrict the cluster communication-wise.

#

#单播发现允许显示的控制哪些节点将被用来发现集群。它可以用在当多播不生效或者限制集群的通信

# 1. Disable multicast discovery (enabled by default):

#

#1.禁止多播发现（默认启用）

#discovery.zen.ping.multicast.enabled: false

#

# 2. Configure an initial list of master nodes in the cluster

#    to perform discovery when new nodes (master or data) are started:

#

#2.在集群中配置一个初始主节点列表来执行发现，当新的节点（主或者数据）开启的时候

#discovery.zen.ping.unicast.hosts: ["host1", "host2:port"]

# EC2 discovery allows to use AWS EC2 API in order to perform discovery.

#

#EC2发现允许使用AWS WC2 API来执行发现

# You have to install the cloud-aws plugin for enabling the EC2 discovery.

#

#必须安装cloud-aws插件启用EC2的发现

# For more information, see

# <http://elasticsearch.org/guide/en/elasticsearch/reference/current/modules-discovery-ec2.html>

#

# See <http://elasticsearch.org/tutorials/elasticsearch-on-ec2/>

# for a step-by-step tutorial.

# GCE discovery allows to use Google Compute Engine API in order to perform discovery.

#

#GCE发现允许使用谷歌计算引擎API来执行发现

# You have to install the cloud-gce plugin for enabling the GCE discovery.

#

#必须安装cloud-gce插件启用GCE发现

# For more information, see <https://github.com/elasticsearch/elasticsearch-cloud-gce>.

# Azure discovery allows to use Azure API in order to perform discovery.

#

#Azure发现允许使用Azure API来执行发现

# You have to install the cloud-azure plugin for enabling the Azure discovery.

#

#必须安装cloud-azure插件启用Azure的发现

# For more information, see <https://github.com/elasticsearch/elasticsearch-cloud-azure>.

## **十三、Slow Log**

################################## Slow Log ##################################

############################ 日志 ##################################

# Shard level query and fetch threshold logging.

#index.search.slowlog.threshold.query.warn: 10s

#index.search.slowlog.threshold.query.info: 5s

#index.search.slowlog.threshold.query.debug: 2s

#index.search.slowlog.threshold.query.trace: 500ms

#index.search.slowlog.threshold.fetch.warn: 1s

#index.search.slowlog.threshold.fetch.info: 800ms

#index.search.slowlog.threshold.fetch.debug: 500ms

#index.search.slowlog.threshold.fetch.trace: 200ms

#index.indexing.slowlog.threshold.index.warn: 10s

#index.indexing.slowlog.threshold.index.info: 5s

#index.indexing.slowlog.threshold.index.debug: 2s

#index.indexing.slowlog.threshold.index.trace: 500ms

################################## GC Logging ################################

#monitor.jvm.gc.young.warn: 1000ms

#monitor.jvm.gc.young.info: 700ms

#monitor.jvm.gc.young.debug: 400ms

#monitor.jvm.gc.old.warn: 10s

#monitor.jvm.gc.old.info: 5s

#monitor.jvm.gc.old.debug: 2s

################################## Security ################################

# Uncomment if you want to enable JSONP as a valid return transport on the

# http server. With this enabled, it may pose a security risk, so disabling

# it unless you need it is recommended (it is disabled by default).

#

#http.jsonp.enable: true