ES数据库安装-配置

## 安装环境

需要jdk版本大于1.8,而且需要后期发布的1.8，早期的1.8可能存在内存指针压缩问题

Centos 版本大于7

ElasticSearch 6.X

Elasticsearch Head (依赖与nodejs，nodejs需要安装grunt 、grunt-cli)

Kibana插件（kibana版本需要和ElasticSearch版本保持一致，否则kibana会报错）

Elasticsearch安装

#### Elasticsearch 5.X集群多节点角色配置

参考文档：<https://blog.csdn.net/laoyang360/article/details/78290484>

ES2.x及之前版本节点角色清单

1、[Master-eligible node](https://www.elastic.co/guide/en/elasticsearch/reference/5.0/modules-node.html#master-node) （符合主节点的节点/主合格节点）

2、[Data node](https://www.elastic.co/guide/en/elasticsearch/reference/5.0/modules-node.html#data-node) （数据节点）

3、[Client node](https://www.elastic.co/guide/en/elasticsearch/reference/5.0/ingest.html) （客户端节点）

ES5.x节点角色清单

1、[Master-eligible node](https://www.elastic.co/guide/en/elasticsearch/reference/5.0/modules-node.html#master-node)（符合主节点的节点/主合格节点）

2、[Data node](https://www.elastic.co/guide/en/elasticsearch/reference/5.0/modules-node.html#data-node) （数据节点）

3、[Ingest node](https://www.elastic.co/guide/en/elasticsearch/reference/5.0/ingest.html) （摄取节点）

4、Coordinating only node （仅协调节点）

[Tribe node](https://www.elastic.co/guide/en/elasticsearch/reference/5.0/modules-tribe.html) （部落节点） 是一种特殊仅协调节点

生产环境节点组合类型的抉择

**Elasticsearch的员工 Christian\_Dahlqvist解读**如下：

一个节点的缺省配置是：主节点+数据节点两属性为一身。对于3-5个节点的小集群来讲，通常让所有节点存储数据和具有获得主节点的资格。你可以将任何请求发送给任何节点，并且由于所有节点都具有集群状态的副本，它们知道如何路由请求。

通常只有较大的集群才能开始分离专用主节点、数据节点。 对于许多用户场景，路由节点根本不一定是必需的。

专用协调节点（也称为client节点或路由节点）从数据节点中消除了聚合/查询的请求解析和最终阶段，并允许他们专注于处理数据。   
在多大程度上这对集群有好处将因情况而异。 通常我会说，在查询大量使用情况下路由节点更常见。

我的思考如下：   
1）对于Ingest节点，如果我们没有格式转换、类型转换等需求，直接设置为false。   
2）3-5个节点属于轻量级集群，要保证主节点个数满足((节点数/2)+1)。   
3）轻量级集群，节点的多重属性如：Master&Data设置为同一个节点可以理解的。   
4）如果进一步优化，5节点可以将Master和Data再分离。

#### 验证内存锁是否开启

方式一：

在命令行中执行

$ curl -XGET '192.168.96.87:9203/\_nodes?filter\_path=\*\*.mlockall&pretty'

方式二：

在浏览器地址栏执行

http://192.168.96.87:9203/\_nodes?filter\_path=\*\*.mlockall&pretty

方式三：

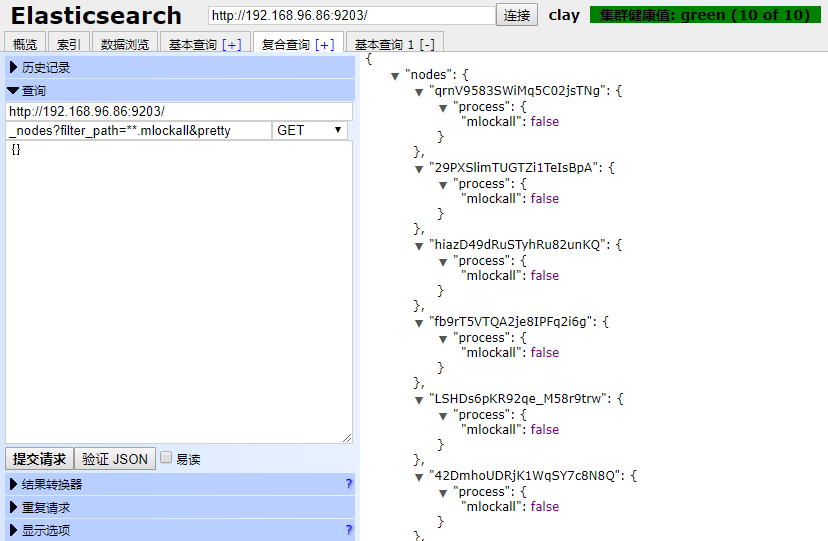
在kibana中执行

切换到kibana的开发工具，打开console，控制台中自动把cURL转换为kibana能识别的命命令，如在命令行中执行的cURL命令粘贴到kibana中，格式转换如下：

GET /\_nodes?filter\_path=\*\*.mlockall&pretty

方式4：

在head插件中查询，查询方式看如下图片



## Elasticsearch内存分配设置

方式一：

编辑./bin/elasticsearch文件

加上：export ES\_HEAP\_SIZE=10g

方式二：

在elasticsearch启动命令后添加jvm参数

./bin/elasticsearch -Xmx10g -Xms10g

## Elasitcsearch-head插件

#### 方式一：开发环境

参考链接：<http://www.51niux.com/?id=201>

1、下载nodejs，这里使用压缩包

wget <https://nodejs.org/dist/v9.3.0/node-v9.3.0-linux-x64.tar.xz>

xz -d node-v9.3.0-linux-x64.tar.xz

tar -xf node-v9.3.0-linux-x64.tar

2、配置环境变量或部署bin文件

NODE\_HOME=$HOME/node

NODE\_HOME=/home/lichangyuan/node

JAVA\_HOME=/home/lichangyuan/sjdk

PATH=$NODE\_HOME/bin:$JAVA\_HOME/bin:$PATH:$HOME/.local/bin:$HOME/bin

export JAVA\_HOME NODE\_HOME PATH

export NODE\_HOME

3、验证nodejs

node -v

npm -v

4、下载head插件

# yum install git -y

# git clone <https://github.com/mobz/elasticsearch-head.git>   #下载head插件文件

5、安装grunt

使用国内淘宝源安装grunt，grunt是基于Node.js的项目构建工具，可以进行打包压缩、测

试、执行等等的工作，head插件就是通过grunt启动

# cd cd elasticsearch-head/

# npm install -g grunt --registry=[https://registry.npm.taobao.org](https://registry.npm.taobao.org/)

npm WARN deprecated coffee-script@1.10.0: CoffeeScript on NPM has moved to "coffeescript" (no hyphen)

/usr/local/node-8.9.1/bin/grunt -> /usr/local/node-8.9.1/lib/node\_modules/grunt/bin/grunt

+ grunt@1.0.1

added 92 packages in 10.604s

6、验证grunt

ls -d node\_modules/grunt

node\_modules/grunt

#如果没产生此目录需要：#cd elasticsearch-head && npm install grunt --save

7、安装Head插件

# npm install -g grunt-cli --registry=https://registry.npm.taobao.org

# npm install  --registry=[https://registry.npm.taobao.org](https://registry.npm.taobao.org/)   #安装head插件

#### 方式二：生产环境

这里是把开发环境的node插件打包了。

nodejs-v9.3.0.tar.gz

elasticsearch-head.tar.gz

解压后需要配置node的环境变量

## Elasticsearch-Kibana安装

下载kibana插件，版本号一定要和elasticsearch版本对应，否则启动报错

## Elasticsearch安全重启

滚动重启

总有一天你会需要做一次集群的滚动重启——保持集群在线和可操作，但是逐一把节点下线。

常见的原因：Elasticsearch 版本升级，或者服务器自身的一些维护操作（比如操作系统升级或者硬件相关）。不管哪种情况，都要有一种特别的方法来完成一次滚动重启。

正常情况下，Elasticsearch 希望你的数据被完全的复制和均衡的分布。如果你手动关闭了一个节点，集群会立刻发现节点的丢失并开始再平衡。如果节点的维护是短期工作的话，这一点就很烦人了，因为大型分片的再平衡需要花费相当的时间（想想尝试复制 1TB 的数据——即便在高速网络上也是不一般的事情了）。

我们需要的是，告诉 Elasticsearch 推迟再平衡，因为对外部因子影响下的集群状态，我们自己更了解。操作流程如下：

1. 可能的话，停止索引新的数据。虽然不是每次都能真的做到，但是这一步可以帮助提高恢复速度。
2. 禁止分片分配。这一步阻止 Elasticsearch 再平衡缺失的分片，直到你告诉它可以进行了。如果你知道维护窗口会很短，这个主意棒极了。你可以像下面这样禁止分配：

PUT /\_cluster/settings

{

"transient" : {

"cluster.routing.allocation.enable" : "none"

}

}

1. 关闭单个节点。

curl -XPOST http://192.168.1.3:9200/\_cluster/nodes/\_local/\_shutdown

1. 执行维护/升级。
2. 重启节点，然后确认它加入到集群了。

./bin/elasticsearch -d

1. 用如下命令重启分片分配：

PUT /\_cluster/settings

{

"transient" : {

"cluster.routing.allocation.enable" : "all"

}

}

分片再平衡会花一些时间。一直等到集群变成 绿色 状态后再继续。

1. 重复第 2 到 6 步操作剩余节点。
2. 到这步你可以安全的恢复索引了（如果你之前停止了的话），不过等待集群完全均衡后再恢复索引，也会有助于提高处理速度。

本地服务器ElastiSearch安装配置

192.168.96.86

es-1 9201 9301 **候选主节点（Master-eligible node）**

es-2 9202 9302 **数据节点（Data node）**

es-3 9203 9303 **客户端节点（Client node）**

192.168.96.87

es-4 9201 9301 **候选主节点（Master-eligible node）**

es-5 9202 9302 **数据节点（Data node）**

es-6 9203 9303 **客户端节点（Client node）**

192.168.96.88

es-7 9201 9301 **候选主节点（Master-eligible node）**

es-8 9202 9302 **数据节点（Data node）**

es-9 9203 9303 **客户端节点（Client node）**

## ElaticSearch的名词和概念

分布式：一个任务分给多台机器去做，减少单个任务的执行时间。

集群：提高单位时间内执行任务数。

分布式：一个任务分给多台机器去做，减少单个任务的执行时间。

集群：提高单位时间内执行任务数。

例如：一个任务由10个子任务组成，每个子任务单独执行需要1个小时，则在一台服务器上执行该任务需要10个小时。

分布式方案：提供10台服务器，每台服务器只处理一个子任务，不考虑任务间的依赖关系，执行完这个任务只需要一个小时。

集群方案：同样提供10台服务器，每台服务器都能独立处理这个任务。假设有10个任务同时到达，10个服务器将同时工作，10小时后，10个任务同时完成，同样是一个小时完成一个任务。

集群概念

集群分类：高可用集群、负载均衡集群、科学计算集群

1、两个关键特性

集群是一组协同工作的服务实体，用以提供比单一服务实体更具扩展性和可用性的服务平台。

可扩展性：集群的性能不限于单一的服务实体，新的服务实体可以动态的加入到集群中

高可用性：同样的服务可以由多个服务实体提供，如果一个服务实体失败了，另一个服务实体会接管失败的服务实体。

2、两大能力：

为了具备高可扩展性和高可用性，要具备两大能力

负载均衡：把任务比较均衡的分配到集群环境下的计算和网络资源

3、两大技术

集群地址

内部通信

错误恢复：由于某种原因，执行某个任务的资源出现故障，另一服务实体中执行同一任务的资源接着完成任务。

分布式 搜索引擎elasticsearch,用来替代solr+zookeeper的集群，都是基于lucene的实现，而lucene是使用java语言开发

而两者都是为数不多的基于json进行索引的搜索引擎。

集群分类：高可用集群、负载均衡集群、科学计算集群

分布式（集群）与集群的联系和区别

分布式是将不同的业务分布在不同的地方

集群是将几台服务器集中在一起，实现同一业务

分布式中每个节点，都可以做集群

而集群不一定是分布式的

## ES 特性和概念

Es特性

接近实时性（NRT）

集群（cluster）

ES 概念

cluster 集群

node 节点，es集群的一个实例。ES的节点角色有那些东西。

index 索引（可以理解为MySQL中的一个逻辑数据库），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，可以给这个文档集合创建客户数据的索引，创建产品目录的所i你，还有订单数据的索引。

一个索引由一个名字来表示（必须全部是小写字符），并且当我们要对某个索引中的文档进行索引，搜索，更新和删除的时候，都要使用这个名字。

type 类型（可以理解为Mysql中的表），在一个索引中，你可以自定义一种或多种类型。一个类型是你的索引的一个逻辑分区上的分类/分区

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。通畅回味具有一组共同字段的文档定义一个类型。比如说，加入我们运行一个博客平台而且将你所有的数据存储到一个索引中。

在这个速印中，你可以为用户数据定义一个类型，微博客数据定义另一个类型，同时也可以为品论数据定义另一个类型

shards 索引分片，把一个完整的索引分成多个分片，这样的好处是可以把一个大的索引拆分成多个。shard有两种类型：原/主（origina/primary)和replica，即主shard以及复制shard。总之，每个索引可以被分成多个分片。一个索引可以被复制0次或者多次，一旦复制了，每个索引就有了主分片和复制分片，

分片和复制的数量可以在索引创建的时候指定。再索引创建之后，你可以在任何时候动态的改变复制的数量，但是事后不能再改变分片的数量

replicas 索引副本(复制) ，shards和replicas一般是同时出现的，

分片和复制概念中的名词，复制源被复制的索引，主分片，复制分片，

recovery 数据恢复或者叫数据重新分布

river es的一个数据源

gateway es的索引快照的存储方式

discovery.zen es的自动发现节点机制

Transport es内部节点或集群与客户端的交互方式

## ES与solr对比

Solr利用Zookeeper进行分布式管理，而ElasticSearch 自身带有分布式协调管理功能。

Solr支持更多格式的数据，而ElasticSearch仅支持JSON文件格式。

ElasticSearch有Query DSL，能够支持更加高级复杂的查询语法，而且可以一次扩展实现类SQL愈发的查询，但是Solr没有Query DSL

Solr 官方提供的功能更多，而Elastic Search本身更注重核心功能，高级功能多由第三方插件提供。

Solr 在传统的搜索应用中表现好于Elastic Search ,但在处理实时应用时效率明显低于Elastic Search

Solr 是传统搜应用的有利解决方案，但是Elastic Search 更适用于新型的实时搜索应用，大数据时代的思考和洞察。

ES安装步骤

安装jdk1.8 用java-version验证

.bash\_profile或 .bashrc

注意：export JAVA\_HOME需要在PATH之前，否则环境变量不能生效

JAVA\_HOME=/home/当前用户/sjdk

PATH=$JAVA\_HOME/bin:$PATH:$HOME/.local/bin:$HOME/bin

export JAVA\_HOME PATH

安装nodejs

可以添加到环境变量

ElasticSearch启动脚本

#!/bin/sh

~/elasticsearch-1/bin/elasticsearch -d

sleep 1

tail -100f ~/es-init/logs-node-1/{cluster.name}.log

# END

elasticsearch.yml配置文件

network.host: 0.0.0.0

# node.master和node.name，共有四种组合， 用来配置节点作为数据节点、主节点、客户端节点,默认都是true,即是主节点也是数据节点，不符合生产环境。

## 这是作为主节点的配置

node.master: true

node.data: false

## 对外提供服务端口

http.port: 9201

## 设置节点间交互的tcp端口

transport.tcp.port: 9301

#discovery.zen.ping.unicast.hosts: ["es87-1:9301", "es87-2:9302", "es87-3:9303"]

## 其他节点的地址端口号，注意端口号为 节点通信端口，不要配置成web发布端口了,可以不配置，会自动发现

discovery.zen.ping.unicast.hosts: ["192.168.96.87:9302","192.168.96.87:9303"]

## 设置集群中自动发现其它节点时ping连接超时时间，默认为3秒,网络比较差可以提高此 值，以防止发现其他节点时出错，发生脑裂。

discovery.zen.ping\_timeout: 6s

## 规则为n/2+1，告诉该节点，其他可以作为主节点的个数

discovery.zen.minimum\_master\_nodes: 1

client.transport.ping\_timeout: 60s

## 允许跨域访问，为了配合elasticsearch-head可视化ES界面

http.cors.enabled: true

## 允许所有地址跨域访问

http.cors.allow-origin: "\*"

path.data: /home/lichangyuan/es-init/data-node-1/01,/home/lichangyuan/es-init/data-node-1/02

path.logs: /home/lichangyuan/es-init/logs-node-1

#END

#节点1

/home/lichangyuan/es-init/data-node-1/01,/home/lichangyuan/es-init/data-node-1/02

/home/lichangyuan/es-init/logs-node-1

#节点2

/home/lichangyuan/es-init/data-node-2/01,/home/lichangyuan/es-init/data-node-2/02

/home/lichangyuan/es-init/logs-node-2

#节点3

/home/lichangyuan/es-init/data-node-3/01,/home/lichangyuan/es-init/data-node-3/02

/home/lichangyuan/es-init/logs-node-3

## 开发模式 vs. 生产模式

默认情况下，Elasticsearch是绑定到本机（network.host: 127.0.0.1），这样es可以进行正常的开发和使用！但是在生产环境下这样是不对的，因为需要外部地址进行发现和被发现从而形成集群！因此，如果Elasticsearch实例不将传输绑定到外部接口（默认），则将Elasticsearch实例置于开发模式，否则将其绑定到外部接口，否则处于生产模式。

## Elasticsearch.yml单播和多播配置

单播配置直接配置discovery.zen.ping.unicast.hosts。组播（广播，多播，群播，多点发送或多点广播）配置需要配置multicast.enabled为true，然后再配置

单播配置

discovery.zen.ping.unicast.hosts: host1:port1, host2:port2

多播配置

discovery.zen.ping.multicast.group: 这是多播请求使用的组地址  
discovery.zen.ping.multicast.port: 这是多播通信使用的端口号  
discovery.zen.ping.multicast.ttl: 指定了经过多少跳之后多播请求将视为无效，默认为3跳  
discovery.zen.ping.multicast.address: 这是elasticsearch应当绑定的地址。默认是null，表示es将会尝试绑定所有可用的网络接口。

要关闭多播功能，需要将multicast.enabled 添加到elasticsearch.yml文件中，并将其值设置为false

## Elasticsearch.yml节点功能配置详解

elasticsearch节点(角色)类型解释node.master和node.data

在生产环境下，如果不修改elasticsearch节点的角色信息，在高数据量，高并发的场景下集群容易出现脑裂等问题。

默认情况下，elasticsearch集群中每个节点都有成为主节点的资格，也都存储数据，还可以提供查询服务。这些功能是由两个属性控制的。

node.master和node.data。默认情况下这两个属性的值都是true。

下面详细介绍一下这两个属性的含义以及不同组合可以达到的效果。node.master：这个属性表示节点是否具有成为主节点的资格注意：此属性的值为true，并不意味着这个节点就是主节点。因为真正的主节点，是由多个具有主节点资格的节点进行选举产生的。所以，这个属性只是代表这个节点是不是具有主节点选举资格。

这两个属性可以有四种组合：

第一种

node.master: true

node.data: true

这种组合表示这个节点即有成为主节点的资格，又存储数据，

这个时候如果某个节点被选举成为了真正的主节点，那么他还要存储数据，这样对于这个节点的压力就比较大了。

elasticsearch默认每个节点都是这样的配置，在测试环境下这样做没问题。实际工作中建议不要这样设置，

这样相当于主节点和数据节点的角色混合到一块了。

第二种

node.master: false

node.data: true

这种组合表示这个节点没有成为主节点的资格，也就不参与选举，只会存储数据。

这个节点我们称为data(数据)节点。在集群中需要单独设置几个这样的节点负责存储数据。后期提供存储和查询服务。

node.master: false

第三种

node.master: true

node.data: false

这种组合表示这个节点不会存储数据，有成为主节点的资格，可以参与选举，有可能成为真正的主节点。

这个节点我们称为master节点

第四种

node.master: false

node.data: false

这种组合表示这个节点即不会成为主节点，也不会存储数据，

这个节点的意义是作为一个client(客户端)节点，主要是针对海量请求的时候可以进行负载均衡。

生产环境配置解析

建议集群中设置3台以上的节点作为master节点【node.master: true node.data: false】

再根据数据量设置一批data节点【node.master: false node.data: true】

这些节点只负责存储数据，后期提供建立索引和查询索引的服务，这样的话如果用户请求比较频繁，这些节点的压力也会比较大

所以在集群中建议再设置一批client节点【node.master: false node.data: true】

这些节点只负责处理用户请求，实现请求转发，负载均衡等功能。

下面配置各种类型节点对服务器资源占用

master节点：普通服务器即可(CPU 内存 消耗一般)

data节点：主要消耗磁盘，内存

client节点：普通服务器即可(如果要进行分组聚合操作的话，建议这个节点内存也分配多一点)

## elasticsearch.yml配置文件详解

**cluster.name: elasticsearch**  
配置es的集群名称，默认是elasticsearch，es会自动发现在同一网段下的es，如果在同一网段下有多个集群，就可以用这个属性来区分不同的集群。

**node.name: "Franz Kafka"**  
节点名，默认随机指定一个name列表中名字，该列表在es的jar包中config文件夹里name.txt文件中，其中有很多作者添加的有趣名字。

**node.master: true**  
指定该节点是否有资格被选举成为node，默认是true，es是默认集群中的第一台机器为master，如果这台机挂了就会重新选举master。

**node.data: true**  
指定该节点是否存储索引数据，默认为true。

**index.number\_of\_shards: 5**  
设置默认索引分片个数，默认为5片。

**index.number\_of\_replicas: 1**  
设置默认索引副本个数，默认为1个副本。

**path.conf: /path/to/conf**  
设置配置文件的存储路径，默认是es根目录下的config文件夹。

**path.data: /path/to/data**  
设置索引数据的存储路径，默认是es根目录下的data文件夹，可以设置多个存储路径，用逗号隔开，例：  
path.data: /path/to/data1,/path/to/data2

**path.work: /path/to/work**  
设置临时文件的存储路径，默认是es根目录下的work文件夹。

**path.logs: /path/to/logs**  
设置日志文件的存储路径，默认是es根目录下的logs文件夹

**path.plugins: /path/to/plugins**  
设置插件的存放路径，默认是es根目录下的plugins文件夹

**bootstrap.mlockall: true**  
设置为true来锁住内存。因为当jvm开始swapping时es的效率会降低，所以要保证它不swap，可以把ES\_MIN\_MEM和ES\_MAX\_MEM两个环境变量设置成同一个值，并且保证机器有足够的内存分配给es。同时也要允许elasticsearch的进程可以锁住内存，linux下可以通过`ulimit -l unlimited`命令。

**network.bind\_host: 192.168.0.1**  
设置绑定的ip地址，可以是ipv4或ipv6的，默认为0.0.0.0。

**network.publish\_host: 192.168.0.1**  
设置其它节点和该节点交互的ip地址，如果不设置它会自动判断，值必须是个真实的ip地址。

**network.host: 192.168.0.1**  
这个参数是用来同时设置bind\_host和publish\_host上面两个参数。

**transport.tcp.port: 9300**  
设置节点间交互的tcp端口，默认是9300。

**transport.tcp.compress: true**  
设置是否压缩tcp传输时的数据，默认为false，不压缩。

**http.port: 9200**  
设置对外服务的http端口，默认为9200。

**http.max\_content\_length: 100mb**  
设置内容的最大容量，默认100mb

**http.enabled: false**  
是否使用http协议对外提供服务，默认为true，开启。

**gateway.type: local**  
gateway的类型，默认为local即为本地文件系统，可以设置为本地文件系统，分布式文件系统，hadoop的HDFS，和amazon的s3服务器，其它文件系统的设置方法下次再详细说。

**gateway.recover\_after\_nodes: 1**  
设置集群中N个节点启动时进行数据恢复，默认为1。

**gateway.recover\_after\_time: 5m**  
设置初始化数据恢复进程的超时时间，默认是5分钟。

**gateway.expected\_nodes: 2**  
设置这个集群中节点的数量，默认为2，一旦这N个节点启动，就会立即进行数据恢复。

**cluster.routing.allocation.node\_initial\_primaries\_recoveries: 4**  
初始化数据恢复时，并发恢复线程的个数，默认为4。

**cluster.routing.allocation.node\_concurrent\_recoveries: 2**  
添加删除节点或负载均衡时并发恢复线程的个数，默认为4。

**indices.recovery.max\_size\_per\_sec: 0**  
设置数据恢复时限制的带宽，如入100mb，默认为0，即无限制。

**indices.recovery.concurrent\_streams: 5**  
设置这个参数来限制从其它分片恢复数据时最大同时打开并发流的个数，默认为5。

**discovery.zen.minimum\_master\_nodes: 1**  
设置这个参数来保证集群中的节点可以知道其它N个有master资格的节点。默认为1，对于大的集群来说，可以设置大一点的值（2-4），

只要启动主节点数等于这个值是，集群再回处于ready状态，否则所以已经启动的节点会报：

not enough master nodes discovered during pinging (found [[]], but needed [1]), pinging again

**discovery.zen.ping.timeout: 3s**  
设置集群中自动发现其它节点时ping连接超时时间，默认为3秒，对于比较差的网络环境可以高点的值来防止自动发现时出错。

**discovery.zen.ping.multicast.enabled: false**  
设置是否打开多播发现节点，默认是true。

discovery.zen.ping.unicast.hosts: ["host1", "host2:port", "host3[portX-portY]"]  
设置集群中master节点的初始列表，可以通过这些节点来自动发现新加入集群的节点。

下面是一些查询时的慢日志参数设置  
index.search.slowlog.level: TRACE  
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

## 在生产环境部署经常出现的错误和问题

### 错误一

max virtual memory areas vm.max\_map\_count [65530] is too low, increase to at least [262144]

需要配置

sudo vi /etc/sysctl.conf

添加下面内容：

vm.max\_map\_count=655360

并执行命令：

sysctl –p

### 错误二

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

添加需要配置

sudo vim /etc/security/limits.conf

添加下面内容：

# elasticsearch config start

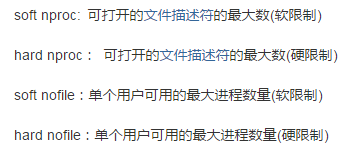
\* soft nofile 65536

\* hard nofile 131072

\* soft nproc 2048

\* hard nproc 4096

# elasticsearch config end



nofile 当前用户最大文件打开数

noprec 当前用户最大创建进程数

**vm.max\_map\_count 一个进程可以拥有的虚拟内存区域数量**

有关最大文件描述符的问题

<https://www.cnblogs.com/taosim/articles/4444518.html>

Linux 文件描述符的问题

Linux下，文件描述符就是一个简单的整数值，习惯上，标准输入（standard input）的文件描述符是 0，标准输出（standard output）是 1，标准错误（standard error）是 2。POSIX 定义了STDIN\_FILENO、STDOUT\_FILENO 和STDERR\_FILENO 来代替 0、1、2。这三个符号常量的定义位于头文件 unistd.h。

**sysctl -w file-max=655350**

 Sysctl也是临时的，要想永久生效，可以通过更改sysctl的文件，编辑/etc/sysctl.conf文件，添加或修改以下一行：

fs.file-max=655350

### 生产环境elasticsearch建议

<https://blog.csdn.net/thomas0yang/article/details/55518105>

ElasticSearch增加机器的配置

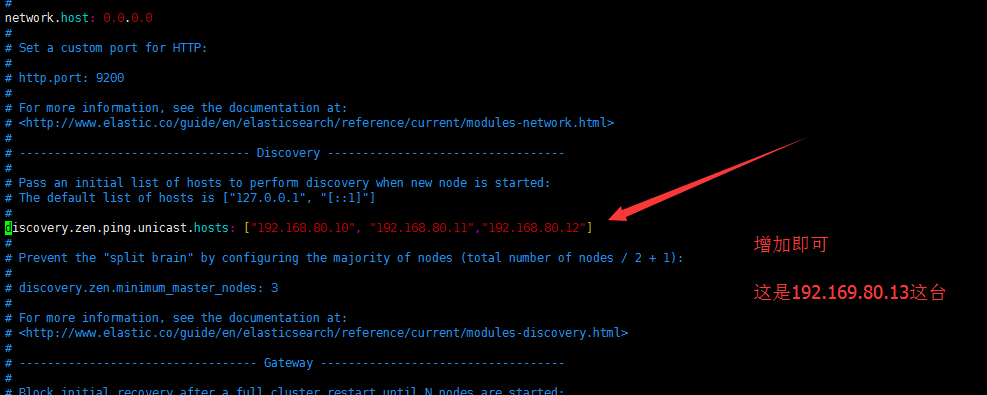
比如，你已经成功搭建了3台机器的es集群，如我这里分别是192.168.80.10、192.168.80.11、192.168.80.12。

那么，**我现在，想增加192.168.80.13到现有的3台es集群，扩容至4台的es集群，如何来操作呢？**

　　解决办法：

只需，在192.168.80.13的这台机器的elasticsearch.yml的，只需让新的这台机器，能找到之前3台的信息即可。

就如，A认识B,B认识C,C认识D，则说A认识D。因为，对于ES而言，没那么严格！



### Kibana和ElasticSearch版本对照问题

案发现场：安装es版本时6.3.0，kibana版本时6.3.2,启动kibana时一直报错

[warning] You're running Kibana 6.3.2 with some different versions of Elasticsearch. Update Kibana or Elasticsearch to the same version to prevent compatibility issues: v6.3.0 @ 192.168.96.87:9203 (192.168.96.87), v6.3.0 @ 192.168.96.87:9201 (192.168.96.87), v6.3.0 @ 192.168.96.87:9202 (192.168.96.87)

### 写后台启动kibana脚本

nohup ./kibana > kibana.out &2> &1 &

解析：

0 – stdin (standard input) 标准输入流，

1 – stdout (standard output) 标准输出流，表示屏幕输出

2 – stderr (standard error) 标准错误流 ；

nohup command &

表示退出帐户/关闭终端之后继续运行相应的进程。Nohup就是不挂起的意思。

nohup command > nohup.out &

表示后台运行，把command的标准输出复制到nohup.out文件。

nohup command > nohup.out 2> &1 &

表示后台运行，把错误输出重定向到标准输出描述符1，标准输出描述符输出到文件。

输出到/dev/null，意思是输出丢进垃圾桶。

tail -f      等同于--follow=descriptor，根据文件描述符进行追踪，当文件改名或被删除，追踪停止

tail -F     等同于--follow=name  --retry，根据文件名进行追踪，并保持重试，即该文件被删除或改名后，如果再次创建相同的文件名，会继续追踪

tailf        等同于tail -f -n 10（貌似tail -f或-F默认也是打印最后10行，然后追踪文件），与tail -f不同的是，如果文件不增长，它不会去访问磁盘文件，所以tailf特别适合那些便携机上跟踪日志文件，因为它减少了磁盘访问，可以省电

### Linux前台、后台、挂起、退出、查看命令汇总

command &  直接在后台运行程序  
ctrl+c 退出前台的命令,不再执行  
ctrl+z挂起前台命令暂停执行，回到shell命令行环境中  
bg    将刚挂起的命令放到后台运行  
bg %3  将第三个job放到后台运行  
kill %3  杀死第三个job，不再执行  
fg    将刚挂起的命令返回前台运行  
fg %3  将第三个job返回前台运行  
jobs   察看当前shell下运行的所有程序；带+表示最新的jobs；带-表示次新的jobs；其他jobs不带符号

nohup=no hang up，不挂断，如果你正在运行一个进程，而且你觉得在退出帐户时该进程还不会结束，那么可以使用nohup命令。该命令可以在你退出帐户/关闭终端之后继续运行相应的进程.长命令必须写在shell文件中，否则nohup不起作用

          nohup command &                 //该命令的一般形式  
          nohup command > myout.file 2>&1 &      //log输出到myout.file，并将标准错误输出重定向到标准输出，再被重定向到myout.file

### 集群部署错误三

单播配置

discovery.zen.ping.unicast.hosts: ["192.168.96.86:9301","192.168.96.87:9301","192.168.96.88:9301","192.168.96.86:9302","192.168.96.87:9302","192.168.96.88:9302","192.168.96.86:9303","192.168.96.87:9303",

"192.168.96.88:9303"]

端口错写为es对外提供服务的restful端口，导致启动后一直找不到足够的主节点。

### 集群部署错误四

粘贴单播列表，发生回行字符，在SecureCRT中粘贴超过一行的字符串，会自带一个会行字符，很隐蔽有时候不容易发现，造成一些很隐晦的错误。众包质检生产环境也出现过这样的问题。