# ES全文检索

## 课前准备

yum install lrzsz wget vim unzip net-tools -y

yum install -y java-1.8.0-openjdk-devel.x86\_64

yum install java-11-openjdk-devel.x86\_64 -y

Jdk的路径

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.252.b09-2.el7\_8.x86\_64

需要安装gcc：yum install gcc-c++

1、下载redis的源码包。

2、把源码包上传到linux服务器

3、解压源码包

tar -zxvf redis-3.0.0.tar.gz

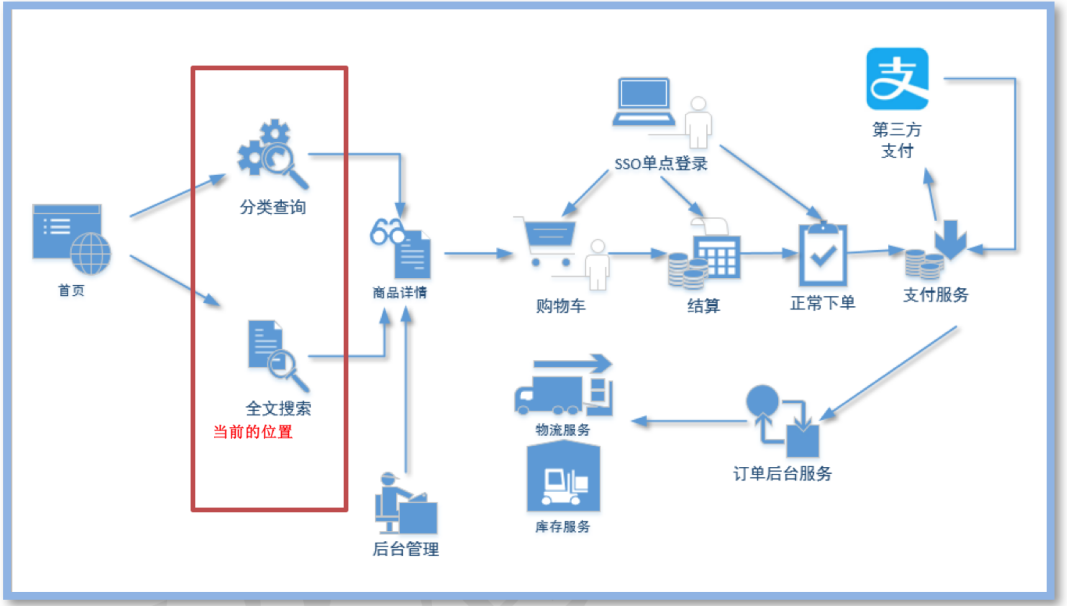
cd redis-3.0.0

4、make

5、make install

6、make install PREFIX=/usr/local/redis

## 场景



## 搜索

### 概念

什么是搜索， 计算机根据用户输入的关键词进行匹配，从已有的数据库中摘录出相关的记录反馈给用户。

常见的全网搜索引擎，像百度、谷歌这样的。但是除此以外，搜索技术在垂直领域也有广泛的使用，比如淘宝、京东搜索商品，万芳、知网搜索期刊，csdn中搜索问题贴。也都是基于海量数据的搜索

### 关系性数据库



弊端：

1.对于传统的关系性数据库对于关键词的查询，只能逐字逐行的匹配，性能非常差。

2.匹配方式不合理，比如搜索"小密手机" ，如果用like进行匹配， 根本匹配不到。但是考虑使用者的用户体验的话，除了完全匹配的记录，还应该显示一部分近似匹配的记录，至少应该匹配到"手机"。

### 倒排索引

全文搜索引擎目前主流的索引技术就是倒排索引的方式

倒排索引的保存数据的方式是

单词→记录

如搜索"红海行动" 数据库中保存格式



那么搜索引擎是如何匹配的呢？

基于分词技术构建倒排索引

首先每个记录保存数据时，都不会直接存入数据库。系统先会对数据进行分词，然后以倒排索引结构保存。如下：



然后等到用户搜索的时候，会把搜索的关键词也进行分词，会

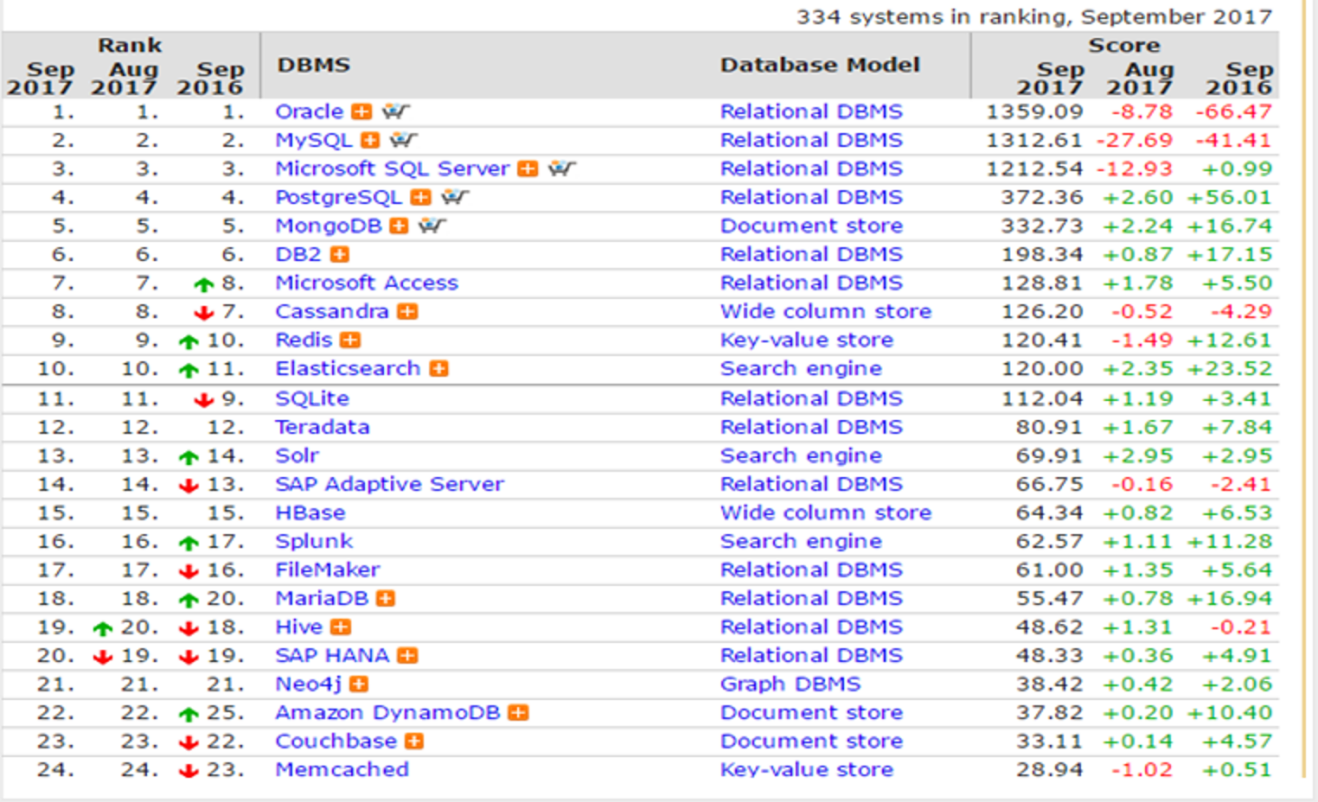
把"红海行动"分词分成：红海和行动两个词

这样的话，先用红海进行匹配，得到id=1和id=2的记录编号，再用行动匹配可以迅速定位id为1,3的记录。

那么全文索引通常，还会根据匹配程度进行打分，显然1号记录能匹配的次数更多。所以显示的时候以评分进行排序的话，1号记录会排到最前面。而2、3号记录也可以匹配到。

## 全文检索工具elasticsearch

### 数据库排行



### lucene与elasticsearch

咱们之前讲的处理分词，构建倒排索引，等等，都是这个叫lucene的做的。那么能不能说这个lucene就是搜索引擎呢？还不能。lucene只是一个提供全文搜索功能类库的核心工具包，而真正使用它还需要一个完善的服务框架搭建起来的应用。好比lucene是类似于jdk，而搜索引擎软件就是tomcat 的。

目前市面上流行的搜索引擎软件，主流的就两款，elasticsearch和solr,这两款都是基于lucene的搭建的，可以独立部署启动的搜索引擎服务软件。由于内核相同，所以两者除了服务器安装、部署、管理、集群以外，对于数据的操作，修改、添加、保存、查询等等都十分类似。就好像都是支持sql语言的两种数据库软件。只要学会其中一个另一个很容易上手。

从实际企业使用情况来看，elasticSearch的市场份额逐步在取代solr，国内百度、京东、新浪都是基于elasticSearch实现的搜索功能。国外就更多了 像维基百科、GitHub、Stack Overflow等等也都是基于ES的

### ES官网

官方学习首页<https://www.elastic.co/guide/index.html>

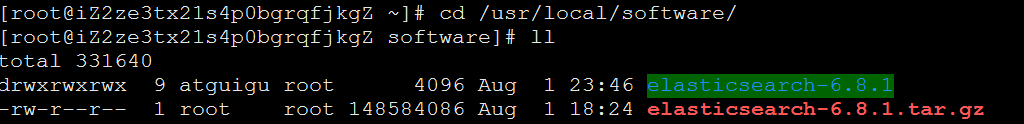
下载首页 <https://www.elastic.co/cn/downloads/>

wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.12.1-linux-x86\_64.tar.gz

wget https://artifacts.elastic.co/downloads/kibana/kibana-7.12.1-linux-x86\_64.tar.gz

## ES的安装

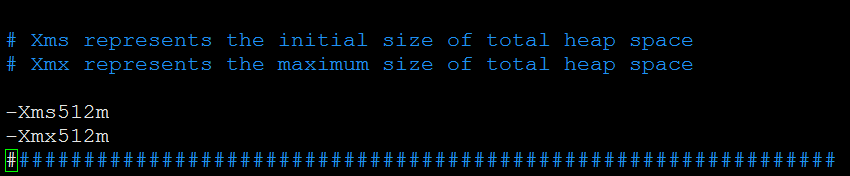
1. 上传安装包到/usr/local/software目录下
2. 解压文件tar -zxvf elasticsearch-7.12.1-linux-x86\_64.tar.gz



1. 切换到目录 修改jvm.options

cd /usr/local/software/elasticsearch-7.12.1/config

vim jvm.options 把1g改小一些



1. 修改配置文件

cd /usr/local/software/elasticsearch-7.12.1/config

vim elasticsearch.yml

添加一下内容

|  |
| --- |
| cluster.name: my-es  node.name: node-1  **network.host: 0.0.0.0**  cluster.initial\_master\_nodes: ["node-1"] |

**修改yml配置的注意事项:**

每行必须顶格，不能有空格

"："后面必须有一个空格

集群名称，同一集群名称必须相同

1. 由于es的安全性需要非root用户启动
2. adduser atguigu passwd atguigu
3. chown -R atguigu elasticsearch-7.12.1

## 修改linux配置

为什么要修改linux配置？

默认elasticsearch是单机访问模式，就是只能自己访问自己。

但是我们之后一定会设置成允许应用服务器通过网络方式访问。这时，

elasticsearch就会因为嫌弃单机版的低端默认配置而报错，甚至无法启动。

所以我们在这里就要把服务器的一些限制打开，能支持更多并发。

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

原因：系统允许 Elasticsearch 打开的最大文件数需要修改成65536

解决：vim /etc/security/limits.conf

添加内容：

|  |
| --- |
| \* soft nofile 65536  \* hard nofile 131072  \* soft nproc 2048  \* hard nproc 65536 |

注意："\*" 不要省略掉

**问题2：max virtual memory areas vm.max\_map\_count [65530] likely too low, increase to at least [262144]**

原因：一个进程可以拥有的虚拟内存区域的数量。

解决：可零时提高vm.max\_map\_count的大小

命令：sysctl -w vm.max\_map\_count=262144

vim /etc/sysctl.conf

vm.max\_map\_count=262144

sysctl -p

**重启Linux**

## 配置环境变量

vim /etc/profile

|  |
| --- |
| export JAVA\_HOME=**/usr/lib/jvm/java**  export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar:$JAVA\_HOME/jre/lib/rt.jar  export PATH=$PATH:$JAVA\_HOME/bin |

source /etc/profile

**启动**

## 启动

切换到目录 /usr/local/software/elasticsearch-7.12.1

bin/elasticsearch (-d) 非root用户执行

1. 核心配置文件

/etc/elasticsearch/elasticsearch.yml

数据文件路径

/var/lib/elasticsearch/

日志文件路径

/var/log/elasticsearch/elasticsearch.log

## 测试

curl http://182.92.234.71:9200/

浏览器 http://182.92.234.71:9200/(防火墙)

systemctl disable firewalld

systemctl stop firewalld

如果启动未成功(锻炼自己的能力)

如果启动未成功，请去查看相关日志

vim /usr/local/software/elasticsearch-7.12.1/logs/my-es.log

## 安装kibana

1.解压文件

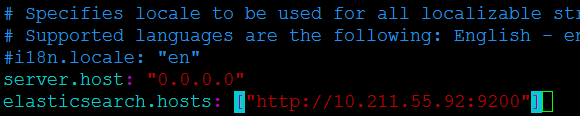
tar -zxvf kibana-7.12.1-linux-x86\_64.tar.gz

2.切换目录修改配置文件

cd /usr/local/software/kibana-7.12.1-linux-x86\_64/config

vim kibana.yml

|  |
| --- |
| server.host: "0.0.0.0"  elasticsearch.hosts: ["http://10.211.55.95:9200"] |



3.启动kibana

前台启动(切换到kibana的bin目录)

bin/kibana --allow-root

后台启动

nohup bin/kibana --allow-root &

4.打开控制台

http://182.92.234.71:5601

在DevTools的Console中

执行get \_cluster/health

## elasticsearch的基本概念

|  |  |
| --- | --- |
| cluster | 整个elasticsearch 默认就是集群状态，整个集群是一份完整、互备的数据。 |
| node | 集群中的一个节点，一般只一个进程就是一个node |
| shard | 分片，即使是一个节点中的数据也会通过hash算法，分成多个片存放，默认是5片。高可靠 |
| index | 相当于rdbms的database, 对于用户来说是一个逻辑数据库，虽然物理上会被分多个shard存放，也可能存放在多个node中。 |
| type | 类似于rdbms的table，但是与其说像table，其实更像面向对象中的class , 同一Json的格式的数据集合。 |
| document | 类似于rdbms的 row、面向对象里的object |
| field | 相当于字段、属性 |



# kibana学习elasticsearch restful api (DSL)

### ES数据结构

Java对象结构

|  |
| --- |
| public class Movie {  String id;  String name;  Double doubanScore;  List<Actor> actorList;  }  public class Actor{  String id;  String name;  } |

这两个对象如果放在关系型数据库保存，会被拆成2张表，但是elasticsearch是用一个json来表示一个document。

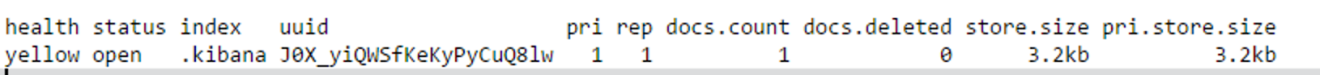
所以es中保存的数据结构长下面那个样子：

|  |
| --- |
| {  "id":"1",  "name":"operation red sea",  "doubanScore":"8.5",  "actorList":[  {"id":"1","name":"zhangyi"},  {"id":"2","name":"haiqing"},  {"id":"3","name":"zhanghanyu"}  ]  } |

### ES基本命令

查看es中有哪些索引

GET /\_cat/indices?v



es 中会默认存在一个名为.kibana的索引

表头的含义

|  |  |
| --- | --- |
| health | green(集群完整) yellow(单点正常、集群不完整) red(单点不正常) |
| status | 是否能使用 |
| index | 索引名 |
| uuid | 索引统一编号 |
| pri | 主节点几个 |
| rep | 从节点几个 |
| docs.count | 文档数 |
| docs.deleted | 文档被删了多少 |
| store.size | 整体占空间大小 |
| pri.store.size | 主节点占 |

### 对数据的操作

#### 增加一个索引

PUT /movie\_index



#### 删除一个索引

ES是不删除也不修改任何数据

|  |
| --- |
| DELETE /movie\_index |

#### 新增文档

格式 PUT /index/type/**id**

|  |
| --- |
| PUT /movie\_index/movie/1  { "**id**":1,  "name":"operation red sea",  "doubanScore":8.5,  "actorList":[  {"id":1,"name":"zhang yi"},  {"id":2,"name":"hai qing"},  {"id":3,"name":"zhang han yu"}  ]  }  PUT /movie\_index/movie/2  {  "id":2,  "name":"operation meigong river",  "doubanScore":8.0,  "actorList":[  {"id":3,"name":"zhang han yu"}  ]  }  PUT /movie\_index/movie/3  {  "id":3,  "name":"incident red sea",  "doubanScore":5.0,  "actorList":[  {"id":4,"name":"zhang fei"}  ]  } |

#### 直接用id查找

GET movie\_index/movie/1

#### 修改—整体替换

和新增没有区别

|  |
| --- |
| PUT /movie\_index/movie/3  {  "doubanScore": "7.0"  } |

#### 修改—某个字段

先恢复id为3的记录 再执行下面操作

|  |
| --- |
| POST movie\_index/movie/3/\_update  {  "doc": {  "doubanScore":"7.0"  }  } |

#### 删除一个document

DELETE movie\_index/movie/3

#### 搜索type全部数据

GET movie\_index/movie/\_search



#### 按条件查询(全部)

|  |
| --- |
| GET movie\_index/movie/\_search  {  "query":{  "match\_all": {}  }  } |

#### 按分词查询

|  |
| --- |
| GET movie\_index/movie/\_search  {  "query":{  "match": {"name":"red"}  }  } |

<https://www.cnblogs.com/forfuture1978/archive/2010/03/07/1680007.html>

竞价排名

#### 按分词子属性查询

|  |
| --- |
| GET movie\_index/movie/\_search  {  "query":{  "match": {"actorList.name":"fei"}  }  } |

#### match phrase(词组查询)

|  |
| --- |
| GET movie\_index/movie/\_search  {  "query":{  "match\_phrase": {"name":"operation red"}  }  } |

按短语查询，不再利用分词技术，直接用短语在原始数据中匹配

#### 关键字查询(对查询条件不进行分词)

|  |
| --- |
| GET movie\_index/movie/\_search  {  "query":{  "match": {  "name.keyword": "operation red sea"  }  }  } |



#### fuzzy查询(模糊查询)

|  |
| --- |
| GET movie\_index/movie/\_search  {  "query":{  "fuzzy": {"name":"rad"}  }  } |

校正匹配分词，当一个单词都无法准确匹配，es通过一种算法对非常接近的单词也给与一定的评分，能够查询出来，但是消耗更多的性能。

#### 过滤--查询后过滤

|  |
| --- |
| GET movie\_index/movie/\_search  {  "query":{  "match": {"name":"red"}  },  "post\_filter":{  "term": {  "actorList.id": 3  }  }  } |

#### 过滤--先过滤再查询（推荐）

|  |
| --- |
| GET movie\_index/movie/\_search  {  "query": {  "bool": {  "filter": [  {  "term": {  "actorList.name.keyword": "zhang han yu"  }  }  ],  "must": {  "match": {  "name": "river"  }  }  }  }  } |

#### 过滤--按范围过滤

|  |
| --- |
| GET movie\_index/movie/\_search  {  "query": {  "bool": {  "filter": {  "range": {  "doubanScore": {"gte": 8}  }  }  }  }  } |

关于范围操作符：

|  |  |
| --- | --- |
| gt | 大于 |
| lt | 小于 |
| gte | 大于等于 |
| lte | 小于等于 |

#### 排序

|  |
| --- |
| GET movie\_index/movie/\_search  {  "query":{  "match": {"name":"red sea"}  }  , "sort": [  {  "doubanScore": {  "order": "desc"  }  }  ]  } |

#### 分页查询

|  |
| --- |
| GET movie\_index/movie/\_search  {  "query": { "match\_all": {} },  "from": 1,  "size": 1  } |

#### 指定查询的字段

|  |
| --- |
| GET movie\_index/movie/\_search  {  "query": { "match\_all": {} },  "\_source": ["name", "doubanScore"]  } |

#### 高亮

<p>最近一个爱国商品<b style="color:red">鸿星尔克</b>质量杠杠的</p>

|  |
| --- |
| GET movie\_index/movie/\_search  {  "query":{  "match": {"name":"red sea"}  },  "highlight": {  "fields": {"name":{} }  }    }  自定义标签 加粗  GET movie\_index/movie/\_search  {  "query": {  "match": {  "name": "red sea"  }  },  "highlight": {  "pre\_tags": [  "<b style='color:red'>"  ],  "post\_tags": [  "</b>"  ],  "fields": {  "name": {}  }  }  } |

#### 聚合

取出每个演员共参演了多少部电影

|  |
| --- |
| GET movie\_index/movie/\_search  {  "aggs": {  "groupby\_actor": {  "terms": {  "field": "actorList.name.keyword"  }  }  }  } |

每个演员参演电影的平均分是多少，并按评分排序

|  |
| --- |
| GET movie\_index/movie/\_search  {  "aggs": {  "groupby\_actor\_id": {  "terms": {  "field": "actorList.name.keyword" ,  "order": {  "avg\_score": "desc"  }  },  "aggs": {  "avg\_score":{  "avg": {  "field": "doubanScore"  }  }  }  }  }  } |

# 映射mapping

之前说type可以理解为table，那每个字段的数据类型是如何定义的呢

也就是说可以把mapping理解为表的结构

### es的mapping结构

GET movie\_index/\_mapping

<https://www.elastic.co/guide/en/elasticsearch/reference/current/dynamic-field-mapping.html>



实际上每个type中的字段是什么数据类型，由mapping定义。

但是如果没有设定mapping系统会自动，根据一条数据的格式来推断出应该的数据格式。

* true/false → boolean
* 1020 → long
* 20.1 → double
* "2018-02-01" → date
* "hello world" → text +keyword

**默认只有text会进行分词，keyword是不会分词的字符串**。

mapping除了自动定义，还可以手动定义，但是只能对新加的、没有数据的字段进行定义。一旦有了数据就无法再做修改了。

注意：虽然每个Field的数据放在不同的type下,但是同一个名字的Field在一个index下只能有一种mapping定义。

默认数字类型不支持自动匹配

PUT my-index-000001

{

"mappings": {

"numeric\_detection": true

}

}

delete my-index-000001

PUT my-index-000001/\_doc/1

{

"my\_float": "1.0",

"my\_integer": "1"

}

PUT my-index-000001/\_doc/1

{

"create\_date": "2015/09/02"

}

GET my-index-000001/\_mapping

# 中文分词

elasticsearch本身自带的中文分词，就是单纯把中文一个字一个字的分开，根本没有词汇的概念。但是实际应用中，用户都是以词汇为条件，进行查询匹配的，如果能够把文章以词汇为单位切分开，那么与用户的查询条件能够更贴切的匹配上，查询速度也更加快速。

分词器下载网址：<https://github.com/medcl/elasticsearch-analysis-ik>

**注意版本需要一致**

**最全分词器**

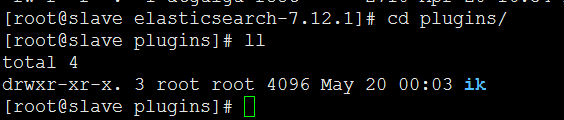
**https://blog.csdn.net/fendouaini/article/details/82027310**

### 安装步骤

1.拷贝IK分词器到ES插件里面

**a**.切换到目录/usr/local/software/elasticsearch-7.12.1/plugins

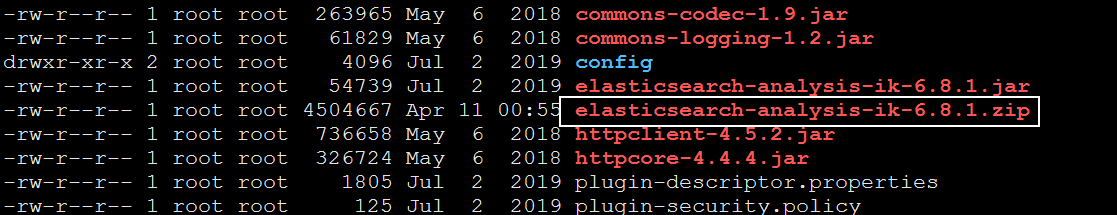
创建目录mkdir ik



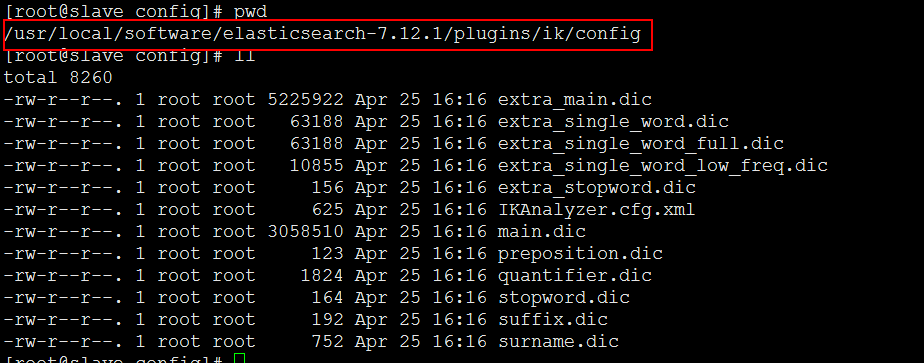
**d**.上传elasticsearch-analysis-ik-7.12.1.zip文件到ik解压

unzip elasticsearch-analysis-ik-7.12.1.zip

再删除ik压缩文件



2.分词器配置里面的结构



1. 非root用户重启es服务

bin/elasticsearch

### 使用

#### 使用默认分词器

|  |
| --- |
| GET movie\_index/\_analyze  {  "text": "迅雷不及掩耳盗铃铃儿响叮当仁不让"  } |

#### 使用分词器

|  |
| --- |
| GET movie\_index/\_analyze  { "analyzer": "ik\_smart",  "text": "迅雷不及掩耳盗铃铃儿响叮当仁不让"  } |

#### 另外一个分词器(ik\_max\_word)

|  |
| --- |
| GET movie\_index/\_analyze  { "analyzer": "ik\_max\_word",  "text": "迅雷不及掩耳盗铃铃儿响叮当仁不让"  } |

能够看出不同的分词器，分词有明显的区别，所以以后定义一个type不能再使用默认的mapping了，要手工建立mapping, 因为要选择分词器

### 基于中文分词搭建索引

#### 建立mapping

https://www.cnblogs.com/feiquan/p/11888812.html

|  |
| --- |
| PUT movie\_chn?include\_type\_name=true  {  "mappings": {  "movie\_type\_chn":{  "properties": {  "id":{  "type": "long"  },  "name":{  "type": "text"  , "analyzer": "ik\_smart"  },  "doubanScore":{  "type": "double"  },  "actorList":{  "properties": {  "id":{  "type":"long"  },  "name":{  "type":"keyword"  }  }  }  }  }  }  } |

#### 插入数据

|  |
| --- |
| PUT /movie\_chn/movie\_type\_chn/1  {  "id": 1,  "name": "红海行动",  "doubanScore": 8.5,  "actorList": [  {  "id": 1,  "name": "张译"  },  {  "id": 2,  "name": "海清"  },  {  "id": 3,  "name": "张含韵"  }  ]  }  PUT /movie\_chn/movie\_type\_chn/2  {  "id": 2,  "name": "湄公河行动",  "doubanScore": 8,  "actorList": [  {  "id": 3,  "name": "张含韵"  }  ]  }  PUT /movie\_chn/movie\_type\_chn/3  {  "id": 3,  "name": "红海事件",  "doubanScore": 5,  "actorList": [  {  "id": 4,  "name": "张飞"  }  ]  } |

#### 查询测试

|  |
| --- |
| GET /movie\_chn/movie\_type\_chn/\_search  {  "query": {  "match": {  "name": "红海战役"  }  }  }  GET /movie\_chn/movie\_type\_chn/\_search  {  "query": {  "term": {  "actorList.name": "张译"  }  }  } |

### 扩展词库

搜索白名单

如用户在京东上搜索"雨女无瓜"马上出来结果

喜大普奔

搜索黑名单

金三胖

修改/usr/local/software/elasticsearch-7.12.1-linux-x86\_64/plugins/elasticsearch/config中的IKAnalyzer.cfg.xml

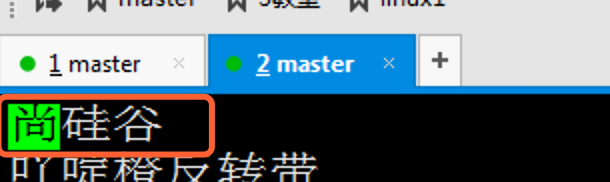
#### 本地扩展

**白名单**

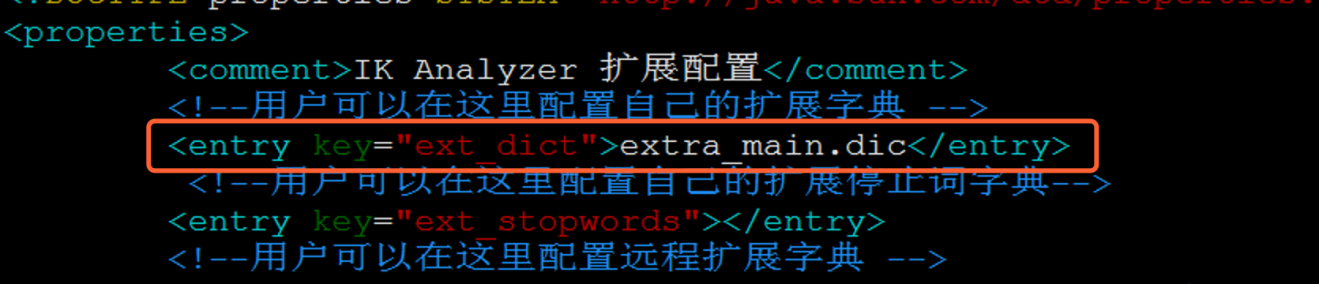
1. 修改扩展词

切换目录/usr/local/software/elasticsearch-7.12.1-linux-x86\_64/plugins/elasticsearch/config

vim extra\_main.dic



1. 修改IKAnalyzer.cfg.xml



1. 重启es服务

service elasticsearch restart

1. 查询测试

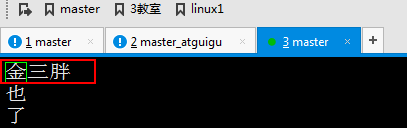
|  |
| --- |
| GET movie\_index/\_analyze  { "analyzer": "ik\_smart",  "text": "尚硅谷"  } |

1. 弊端

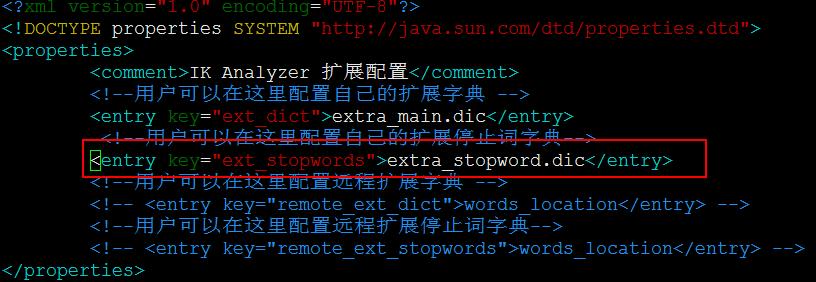
每次修改需要重启ES服务 影响客户体验 不允许中断查询

**黑名单**

1. 修改extra\_stopword.dic



1. 修改IKAnalyzer.cfg.xml



1. 重启es服务

service elasticsearch restart

1. 查询测试

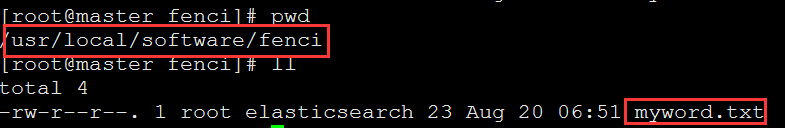
|  |
| --- |
| GET movie\_index/\_analyze  { "analyzer": "ik\_smart",  "text": "尚硅谷"  } |

#### 远程扩展

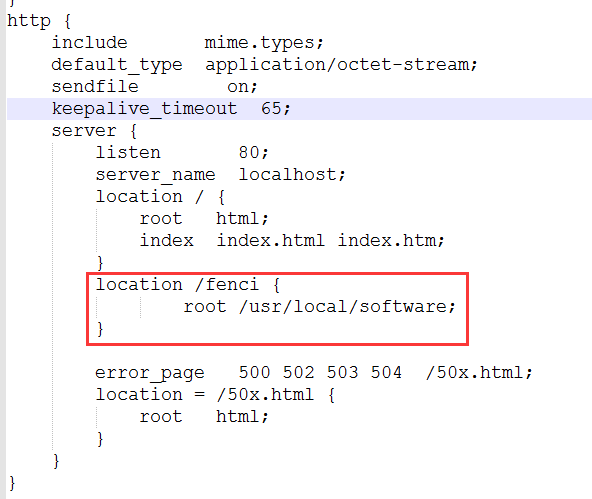
1. 利用nginx搭建静态资源访问服务
2. 在以下目录创建一个

/usr/local/software/fenci

myword.txt



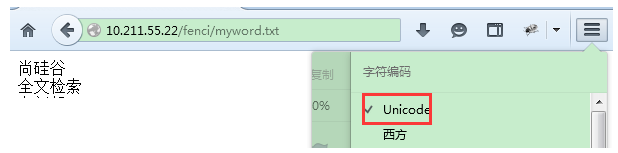
1. 配置nginx.conf文件



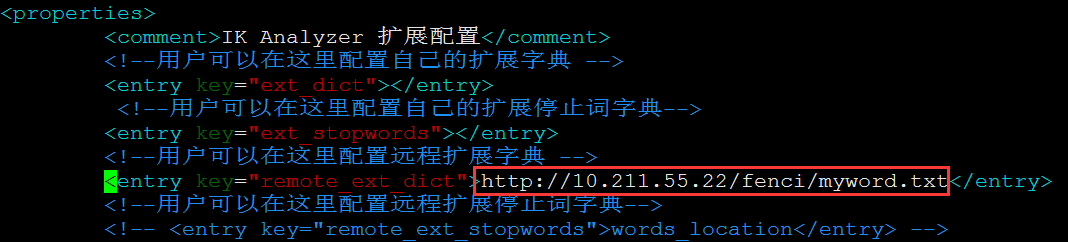
1. 启动nginx

/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf

1. 浏览器测试结果



1. 修改IKAnalyzer.cfg.xml



1. 第一次需要重启ES

service elasticsearch restart

1. 当在myword里面添加信息 需要稍微等一两秒

# Docker安装ES和Kibana(提前下载)

### 安装es

|  |
| --- |
| 1.下载镜像  docker pull docker.elastic.co/elasticsearch/elasticsearch:6.8.1  docker load -i elasticsearch.tar  2.运行  docker run -d -p 9200:9200 -p 9300:9300 --name myes -e "ES\_JAVA\_OPTS=-Xms512m -Xmx512m" -e "discovery.type=single-node" docker.elastic.co/elasticsearch/elasticsearch:6.8.1  3.进入es配置模式  docker exec -it myes /bin/bash  4.启动/停止es容器  docker start myes  docker stop myes  5.启动失败看日志  docker logs -f 73641dd5c7e8  6.删除容器  docker rm myes  7.访问地址  http://172.16.2.128:9200/  8.所有容器  docker ps -a |

### 安装kibana

|  |
| --- |
| 1.下载镜像  docker pull docker.elastic.co/kibana/kibana:6.8.1  docker load -i kibana.tar  2.运行  docker run -d --link myes:elasticsearch --name mykibana -p 5601:5601 docker.elastic.co/kibana/kibana:6.8.1  3.访问地址  http://121.89.208.247:5601  4.删除容器  docker rm mykibana |

### 安装IK

|  |
| --- |
| a.上传ik分词器到linux当中     1. 把ik拷贝到容器里面(需要在plugins里面创建目录ik)   docker start myes  docker exec -it myes /bin/bash  cd /usr/share/elasticsearch/plugins  mkdir ik  exit  docker cp elasticsearch-analysis-ik-6.8.1.zip myes:/usr/share/elasticsearch/plugins/ik  c.解压ik分词器  docker exec -it myes /bin/bash  cd /usr/share/elasticsearch/plugins  unzip elasticsearch-analysis-ik-6.8.1.zip  需要重新启动es容器  **注意要删除elasticsearch-analysis-ik-6.8.1.zip**  d.测试  PUT /movie\_index  GET movie\_index/\_analyze  { "analyzer": "ik\_max\_word",  "text": "迅雷不及掩耳盗铃铃儿响叮当仁不让"  } |