**灰灰商城-ElasticSearch+kibana笔记（尚硅谷谷粒商城2020）够豪横！**

**文章目录**

* [灰灰商城-分布式高级篇-1](https://blog.csdn.net/weixin_44889138/article/details/106895595#1_4)
  + [全文检索-ElasticSearch](https://blog.csdn.net/weixin_44889138/article/details/106895595#ElasticSearch_7)
    - [docker 安装](https://blog.csdn.net/weixin_44889138/article/details/106895595#docker__9)
      * [1、下载镜像文件](https://blog.csdn.net/weixin_44889138/article/details/106895595#1_11)
      * [2、创建实例](https://blog.csdn.net/weixin_44889138/article/details/106895595#2_19)
        + [1. 创建外部ElasticSearch配置文件](https://blog.csdn.net/weixin_44889138/article/details/106895595#1_ElasticSearch_21)
        + [2. 运行 ElasticSearch](https://blog.csdn.net/weixin_44889138/article/details/106895595#2__ElasticSearch_37)
        + [3. 运行kibana](https://blog.csdn.net/weixin_44889138/article/details/106895595#3_kibana_55)
        + [4. 访问 主机+端口](https://blog.csdn.net/weixin_44889138/article/details/106895595#4___66)
      * [3、初步检索](https://blog.csdn.net/weixin_44889138/article/details/106895595#3_91)
        + [1. \_cat](https://blog.csdn.net/weixin_44889138/article/details/106895595#1__cat_93)
        + [2. 索引一个文档（保存）](https://blog.csdn.net/weixin_44889138/article/details/106895595#2__100)
        + [3. 查询文档](https://blog.csdn.net/weixin_44889138/article/details/106895595#3__138)
        + [4. 更新文档](https://blog.csdn.net/weixin_44889138/article/details/106895595#4__174)
        + [5. 删除文档&索引](https://blog.csdn.net/weixin_44889138/article/details/106895595#5__230)
        + [7. 样本测试数据](https://blog.csdn.net/weixin_44889138/article/details/106895595#7__262)
      * [4、进阶检索](https://blog.csdn.net/weixin_44889138/article/details/106895595#4_271)
        + [1. SearchAPI](https://blog.csdn.net/weixin_44889138/article/details/106895595#1_SearchAPI_273)
        + [2. Query DSL -> 查询领域对象语言](https://blog.csdn.net/weixin_44889138/article/details/106895595#2_Query_DSL___306)

[1、一个查询语句的典型结构](https://blog.csdn.net/weixin_44889138/article/details/106895595#1_308)

[2、 match用法](https://blog.csdn.net/weixin_44889138/article/details/106895595#2_match_412)

[3、 match\_phrase -> 短语匹配 (上述match的增强，模糊搜索以短语，不分词)](https://blog.csdn.net/weixin_44889138/article/details/106895595#3_match_phrase___match_439)

[4、 multi\_match 多字段匹配](https://blog.csdn.net/weixin_44889138/article/details/106895595#4_multi_match__495)

[5、 bool 复合查询](https://blog.csdn.net/weixin_44889138/article/details/106895595#5_bool__610)

[6、 filter 结果过滤](https://blog.csdn.net/weixin_44889138/article/details/106895595#6_filter__694)

[7、 term, 与match类似](https://blog.csdn.net/weixin_44889138/article/details/106895595#7_term_match_716)

[8、 aggregations (执行聚合)](https://blog.csdn.net/weixin_44889138/article/details/106895595#8_aggregations__749)

* + - * + [3. Mapping](https://blog.csdn.net/weixin_44889138/article/details/106895595#3_Mapping_909)

[1、字段类型](https://blog.csdn.net/weixin_44889138/article/details/106895595#1_912)

[2、映射（Mapping）](https://blog.csdn.net/weixin_44889138/article/details/106895595#2Mapping_921)

[3、新版本下](https://blog.csdn.net/weixin_44889138/article/details/106895595#3_936)

[4、分词](https://blog.csdn.net/weixin_44889138/article/details/106895595#4_1057)

* + - * [5、安装nginx（为自定义词库创建）](https://blog.csdn.net/weixin_44889138/article/details/106895595#5nginx_1161)
      * [6、Elasticsearch-Rest-Client](https://blog.csdn.net/weixin_44889138/article/details/106895595#6ElasticsearchRestClient_1203)
        + [1、9300:TCP](https://blog.csdn.net/weixin_44889138/article/details/106895595#19300TCP_1205)
        + [2、9200:HTTP](https://blog.csdn.net/weixin_44889138/article/details/106895595#29200HTTP_1211)

**灰灰商城-分布式高级篇-1**

码云地址：https://gitee.com/lin\_g\_g\_hui/grey\_mall

**全文检索-ElasticSearch**

**docker 安装**

**1、下载镜像文件**

docker pull elasticsearch:7.4.2

docker pull kibana:7.4.2-》elasticsearch的可视化工具

**2、创建实例**

**1. 创建外部ElasticSearch配置文件**

mkdir -p /mydata/elasticsearch/config

mkdir -p /mydata/elasticsearch/data

echo "http.host: 0.0.0.0" >> /mydata/elasticsearch/config/elasticsearch.yml

注意： 0.0.0.0之前有一个空格

运行后报错->

Caused by: java.nio.file.AccessDeniedException: /usr/share/elasticsearch/data/nodes"

查看日志-> docker logs elasticsearch

修改权限-任何用户，组可读写执行

chmod -R 777 /mydata/elasticsearch/

**2. 运行 ElasticSearch**

docker run --name elasticsearch -p 9200:9200 -p 9300:9300 \

-e "discovery.type=single-node" \

-e ES\_JAVA\_OPTS="-Xms64m -Xmx512m" \

-v /mydata/elasticsearch/config/elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml \

-v /mydata/elasticsearch/data:/usr/share/elasticsearch/data \

-v /mydata/elasticsearch/plugins:/usr/share/elasticsearch/plugins \

-d elasticsearch:7.4.2

注意：-e ES\_JAVA\_OPTS="-Xms64m -Xmx512m"  
设置占用内存  
查看：  
free -m

**3. 运行kibana**

docker run --name kibana -e ELASTICSEARCH\_URL=http://192.168.72.128:9200 -p 5601:5601 -d kibana:7.4.2

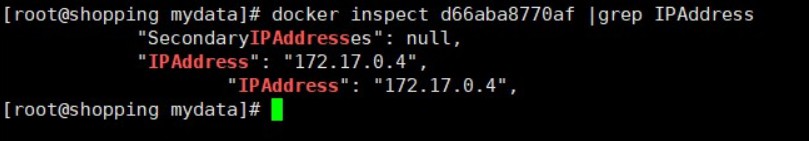
注意主机改为当前的

http://192.168.80.133:9200/

**4. 访问 主机+端口**

http://192.168.80.133:9200/ ->返回json数据则成功  
http://192.168.80.133:9200/\_cat/nodes -> 查看节点

* 注意：使用可视化界面报错，链接ElasticSearch异常  
  http://192.168.80.133:5601/

前面第2点运行时ip为docker中ElasticSearch的IP，查看  
docker inspect d66aba8770af |grep IPAddress  
结果：查找到d66aba8770af的ip为->172.17.0.4  


重新运行：  
docker run --name kibana -e ELASTICSEARCH\_URL=http://172.17.0.4:9200 -p 5601:5601 -d kibana:7.4.2

继续配置：  
进入docker中Kibana文件配置  
docker exec -it kibana /bin/bash  
cd /usr/share/kibana/config/  
vi kibana.yml  
elasticsearch.hosts 改成你es容器的ip，然后将  
xpack.monitoring.ui.container.elasticsearch.enabled 改成 false

**3、初步检索**

**可使用postman 将GET等改为主机+端口**

**1. \_cat**

GET/\_cat/nodes : 查看所有节点  
GET/\_cat/health : 查看es健康状况  
GET/\_cat/master : 查看主节点  
GET/\_cat/indices : 查看所有索引 ==>show databases;

**2. 索引一个文档（保存）**

保存一个数据，保存在哪个索引的哪个类型下，指定用哪个唯一标识  
PUT customer /external/1; 在customer索引下的external类型下保存1号数据为

PUT customer/external/1

1号数据的信息：  
{  
“name”:“Wei-xhh”  
}

PUT和POST都可以

POST新增。如果不指定id, 会自动生成id。指定id就会修改这个数据，并新增版本号

PUT可以新增可以修改。PUT必须指定id；由于PUT需要指定id, 我们一般都用来做修改操作，不指定id会报错。

返回结果：

{

"\_index": "customer",

"\_type": "external",

"\_id": "1",

"\_version": 1,

"result": "created",

"\_shards": {

"total": 2,

"successful": 1,

"failed": 0

},

"\_seq\_no": 0,

"\_primary\_term": 1

}

**3. 查询文档**

GET customer/external/1

返回结果:

{

"\_index": "customer", // 在哪个索引

"\_type": "external", // 在哪个类型

"\_id": "1", // 记录id

"\_version": 2, // 版本号

"\_seq\_no": 1, // 并发控制字段，每次更新就会+1，用来做乐观锁

"\_primary\_term": 1, // 同上，主分片重新分配，如重启，就会变化

"found": true,

"\_source": { // 真正的内容

"name": "Wei-xhh"

}

}

更新携带 ?if\_seq\_no=0 & if\_primary\_term = 1

乐观锁->并发  
1、小明修改1号数据->  
http://192.168.80.133:9200/customer/external/1?if\_seq\_no=0&if\_primary\_term=1  
2、小红修改1号数据->  
http://192.168.80.133:9200/customer/external/1?if\_seq\_no=0&if\_primary\_term=1

情况：  
小明修改了，->成功，对应的seq\_no也自动修改  
小红并不知道已经被小明修改，想要时修改失败 -> 错误码409  
这时小红必须重新查询1号数据得到seq\_no等于什么  
查询后小红得到了seq\_no=5，-> 重新发送请求  
http://192.168.80.133:9200/customer/external/1?if\_seq\_no=5&if\_primary\_term=1  
修改成功。

**4. 更新文档**

POST customer/external/1/\_update

会对比原来数据，与原来一样就什么都不做

{

"doc":{

"name":"wei-xhh6666"

}

}

结果：与原来一样就什么都不做 “result”: “noop”  
\_version，\_seq\_no不变

{

"\_index": "customer",

"\_type": "external",

"\_id": "1",

"\_version": 5,

"result": "noop",

"\_shards": {

"total": 0,

"successful": 0,

"failed": 0

},

"\_seq\_no": 7,

"\_primary\_term": 1

}

或者

POST customer/external/1  
不会检查原来的数据

{

"name":"wei-xhh666"

}

或者

PUT customer/external/1  
不会检查原来的数据

{

"name":"wei-xhh66"

}

更新时也可以同时添加属性

**5. 删除文档&索引**

DELETE customer/external/1  
DELETE customer

1. bulk 批量API

POST customer/external/ \_bulk

{"index":{"\_id":"1"}}

{"name":"wei-xhh"}

{"index":{"\_id":"2"}}

{"name":"wei-xhh66"}

语法格式：

{action: {metadata}}\n

{request body} \n

{action: {metadata}}\n

{request body} \n

复杂实例

POST / \_bulk

{ "delete":{ "\_index":"website", "\_type":"blog", "\_id":"123"}}

{ "create":{ "\_index":"website", "\_type":"blog", "\_id":"123"}}

{ "title":"my first blog post"}

{ "index":{ "\_index":"website", "\_type":"blog"}}

{ "title":"my second blog post"}

{ "update":{ "\_index":"website", "\_type":"blog", "\_id":"123"}}

{ "doc":{ "title":"my updated blog post"}}

**7. 样本测试数据**

https://raw.githubusercontent.com/elastic/elasticsearch/master/docs/src/test/resources/accounts.json  
可能访问不通

**我有数据，如果找不到访问不了可以私我**

POST /bank/account/\_bulk

**4、进阶检索**

**1. SearchAPI**

ES支持两种基本方式检索：

* 一个是通过使用 REST request URI 发送搜索参数 (uri+检索参数）
* 另外一个是通过使用 REST request body 来发送它们 (url+请求体）

设置关机自动重启

docker update 容器id --restart=always

第一种方式：

GET bank/\_search?q=\*&sort=account\_number:asc

第二种方式：

GET bank/\_search

{

"query": {

"match\_all": {}

},

"sort": [

{

"account\_number": "asc"

},

{

"balance": "desc"

}

]

}

**2. Query DSL -> 查询领域对象语言**

**1、一个查询语句的典型结构**

{

QUERY\_NAME:{

ARGUMENT:VALUE,

ARGUMENT:VALUE,...

}

}

* 如果针对某个字段，那么他的结构如下：

{

QUERY\_NAME:{

FIELD\_NAME:{

ARGUMENT:VALUE,

ARGUMENT:VALUE,...

}

}

}

例子

GET bank/\_search

{

"query": {"match\_all": {}},

"sort": [

{

"balance": {

"order": "asc"

}

}

],

"from": 5,

"size": 3,

"\_source": ["balance","age"] // 指定值

}

结果：

{

"took" : 1,

"timed\_out" : false,

"\_shards" : {

"total" : 1,

"successful" : 1,

"skipped" : 0,

"failed" : 0

},

"hits" : {

"total" : {

"value" : 1000,

"relation" : "eq"

},

"max\_score" : null,

"hits" : [

{

"\_index" : "bank",

"\_type" : "account",

"\_id" : "749",

"\_score" : null,

"\_source" : {

"balance" : 1249,

"age" : 36

},

"sort" : [

1249

]

},

{

"\_index" : "bank",

"\_type" : "account",

"\_id" : "402",

"\_score" : null,

"\_source" : {

"balance" : 1282,

"age" : 32

},

"sort" : [

1282

]

},

{

"\_index" : "bank",

"\_type" : "account",

"\_id" : "315",

"\_score" : null,

"\_source" : {

"balance" : 1314,

"age" : 33

},

"sort" : [

1314

]

}

]

}

}

**2、 match用法**

精确查询

GET bank/\_search

{

"query": {

"match": {

"account\_number": "20"

}

}

}

模糊查询 -> 分词匹配

GET bank/\_search

{

"query": {

"match": {

"address": "Kings"

}

}

}

**3、 match\_phrase -> 短语匹配 (上述match的增强，模糊搜索以短语，不分词)**

//不分词匹配

GET bank/\_search

{

"query": {

"match\_phrase": {

"address": "mill lane"

}

}

}

结果：

{

"took" : 1058,

"timed\_out" : false,

"\_shards" : {

"total" : 1,

"successful" : 1,

"skipped" : 0,

"failed" : 0

},

"hits" : {

"total" : {

"value" : 1,

"relation" : "eq"

},

"max\_score" : 9.507477,

"hits" : [

{

"\_index" : "bank",

"\_type" : "account",

"\_id" : "136",

"\_score" : 9.507477,

"\_source" : {

"account\_number" : 136,

"balance" : 45801,

"firstname" : "Winnie",

"lastname" : "Holland",

"age" : 38,

"gender" : "M",

"address" : "198 Mill Lane",

"employer" : "Neteria",

"email" : "winnieholland@neteria.com",

"city" : "Urie",

"state" : "IL"

}

}

]

}

}

**4、 multi\_match 多字段匹配**

GET bank/\_search

{

"query": {

"multi\_match": {

"query": "mill movice",

"fields": ["address","city"]

}

}

}

结果：

{

"took" : 12,

"timed\_out" : false,

"\_shards" : {

"total" : 1,

"successful" : 1,

"skipped" : 0,

"failed" : 0

},

"hits" : {

"total" : {

"value" : 4,

"relation" : "eq"

},

"max\_score" : 5.4032025,

"hits" : [

{

"\_index" : "bank",

"\_type" : "account",

"\_id" : "970",

"\_score" : 5.4032025,

"\_source" : {

"account\_number" : 970,

"balance" : 19648,

"firstname" : "Forbes",

"lastname" : "Wallace",

"age" : 28,

"gender" : "M",

"address" : "990 Mill Road",

"employer" : "Pheast",

"email" : "forbeswallace@pheast.com",

"city" : "Lopezo",

"state" : "AK"

}

},

{

"\_index" : "bank",

"\_type" : "account",

"\_id" : "136",

"\_score" : 5.4032025,

"\_source" : {

"account\_number" : 136,

"balance" : 45801,

"firstname" : "Winnie",

"lastname" : "Holland",

"age" : 38,

"gender" : "M",

"address" : "198 Mill Lane",

"employer" : "Neteria",

"email" : "winnieholland@neteria.com",

"city" : "Urie",

"state" : "IL"

}

},

{

"\_index" : "bank",

"\_type" : "account",

"\_id" : "345",

"\_score" : 5.4032025,

"\_source" : {

"account\_number" : 345,

"balance" : 9812,

"firstname" : "Parker",

"lastname" : "Hines",

"age" : 38,

"gender" : "M",

"address" : "715 Mill Avenue",

"employer" : "Baluba",

"email" : "parkerhines@baluba.com",

"city" : "Blackgum",

"state" : "KY"

}

},

{

"\_index" : "bank",

"\_type" : "account",

"\_id" : "472",

"\_score" : 5.4032025,

"\_source" : {

"account\_number" : 472,

"balance" : 25571,

"firstname" : "Lee",

"lastname" : "Long",

"age" : 32,

"gender" : "F",

"address" : "288 Mill Street",

"employer" : "Comverges",

"email" : "leelong@comverges.com",

"city" : "Movico",

"state" : "MT"

}

}

]

}

}

**5、 bool 复合查询**

复合查询可以合并任何其他查询语句，包括复合语句，这就意味则，复合语句之间可以互相嵌套，可以表达非常复杂的逻辑

GET bank/\_search

{

"query": {

"bool": {

"must": [

{

"match": {

"gender": "F"

}

},

{

"match": {

"address": "mill"

}

}

],

"must\_not": [

{

"match": {

"age": "18"

}

}

],

"should": [

{

"match": {

"lastname": "Wallace"

}

}

]

}

}

}

结果：

{

"took" : 109,

"timed\_out" : false,

"\_shards" : {

"total" : 1,

"successful" : 1,

"skipped" : 0,

"failed" : 0

},

"hits" : {

"total" : {

"value" : 1,

"relation" : "eq"

},

"max\_score" : 6.1104345,

"hits" : [

{

"\_index" : "bank",

"\_type" : "account",

"\_id" : "472",

"\_score" : 6.1104345,

"\_source" : {

"account\_number" : 472,

"balance" : 25571,

"firstname" : "Lee",

"lastname" : "Long",

"age" : 32,

"gender" : "F",

"address" : "288 Mill Street",

"employer" : "Comverges",

"email" : "leelong@comverges.com",

"city" : "Movico",

"state" : "MT"

}

}

]

}

}

**6、 filter 结果过滤**

并不是所有的查询都需要产生分数，特别是那些仅用于filtering (过滤)的文档，为了不计算分数Elasticsearch会自动检查场景并且优化查询的执行。

GET bank/\_search

{

"query": {

"bool": {

"filter": {

"range": {

"age": {

"gte": 19,

"lte": 30

}

}

}

}

}

}

**7、 term, 与match类似**

模糊检索推荐使用match -> 文本字段使用  
精确检索推荐使用term -> 非文本字段使用

GET bank/\_search

{

"query": {

"term": {

"age": "28"

}

}

}

match的精确查找

GET bank/\_search

{

"query": {

"match": {

"address.keyword": "789 Madison Street"

}

}

}

**8、 aggregations (执行聚合)**

聚合提供了从数据中分组和提取数据的能力，最简单的聚合方法大致等于 SQL GROUP BY 和 SQL聚合函数。在Elasticsearch中，您有执行搜索返回hits(命中结果)，并且同时返回聚合结果，把一个响应的所有hits(命中结构)分隔开的能力。可以执行查询和多个聚合，并且在一次使用中得到各自的(任何一个的)返回结果，使用一次简洁和简化的API来避免网络往返。

* 搜索address中包含mill的所有人的年龄分布以及平均年龄，但不显示这些人的详情。

GET bank/\_search

{

"query": {

"match": {

"address": "mill"

}

},

"aggs": {

"ageAgg": {

"terms": {

"field": "age",

"size": 10

}

},

"ageAvg": {

"avg": {

"field": "age"

}

}

}

}

结果

{

"took" : 4643,

"timed\_out" : false,

"\_shards" : {

"total" : 1,

"successful" : 1,

"skipped" : 0,

"failed" : 0

},

"hits" : {

"total" : {

"value" : 4,

"relation" : "eq"

},

"max\_score" : 5.4032025,

"hits" : [

{

"\_index" : "bank",

"\_type" : "account",

"\_id" : "970",

"\_score" : 5.4032025,

"\_source" : {

"account\_number" : 970,

"balance" : 19648,

"firstname" : "Forbes",

"lastname" : "Wallace",

"age" : 28,

"gender" : "M",

"address" : "990 Mill Road",

"employer" : "Pheast",

"email" : "forbeswallace@pheast.com",

"city" : "Lopezo",

"state" : "AK"

}

},

{

"\_index" : "bank",

"\_type" : "account",

"\_id" : "136",

"\_score" : 5.4032025,

"\_source" : {

"account\_number" : 136,

"balance" : 45801,

"firstname" : "Winnie",

"lastname" : "Holland",

"age" : 38,

"gender" : "M",

"address" : "198 Mill Lane",

"employer" : "Neteria",

"email" : "winnieholland@neteria.com",

"city" : "Urie",

"state" : "IL"

}

},

{

"\_index" : "bank",

"\_type" : "account",

"\_id" : "345",

"\_score" : 5.4032025,

"\_source" : {

"account\_number" : 345,

"balance" : 9812,

"firstname" : "Parker",

"lastname" : "Hines",

"age" : 38,

"gender" : "M",

"address" : "715 Mill Avenue",

"employer" : "Baluba",

"email" : "parkerhines@baluba.com",

"city" : "Blackgum",

"state" : "KY"

}

},

{

"\_index" : "bank",

"\_type" : "account",

"\_id" : "472",

"\_score" : 5.4032025,

"\_source" : {

"account\_number" : 472,

"balance" : 25571,

"firstname" : "Lee",

"lastname" : "Long",

"age" : 32,

"gender" : "F",

"address" : "288 Mill Street",

"employer" : "Comverges",

"email" : "leelong@comverges.com",

"city" : "Movico",

"state" : "MT"

}

}

]

},

"aggregations" : {

"ageAgg" : {

"doc\_count\_error\_upper\_bound" : 0,

"sum\_other\_doc\_count" : 0,

"buckets" : [

{

"key" : 38,

"doc\_count" : 2

},

{

"key" : 28,

"doc\_count" : 1

},

{

"key" : 32,

"doc\_count" : 1

}

]

},

"ageAvg" : {

"value" : 34.0

},

"balanceAvg" : {

"value" : 25208.0

}

}

}

可嵌套聚合（道理类似）

**3. Mapping**

**1、字段类型**

7.0版本可有可无  
未来8.0将取消。  
直接将文档存在某个索引下。  
去掉type就是为了提高ES处理数据的效率。

**2、映射（Mapping）**

Mapping是用来定义一个文档，以及它所包含的属性是如何存储和索引的。  
比如使用mapping来定义。

* 哪些字符串属性应该被看做全文本属性。
* 哪些属性包含数字，日期或者地理位置。
* 文档中的所有属性是否都能被索引
* 日期的格式
* 自定义映射规则来执行动态添加属性。
* 查看mapping信息：
  + GET bank/\_mapping
* 修改mapping信息

**3、新版本下**

* 创建映射

规定 my\_index 这个索引下的属性类型

PUT /my\_index

{

"mappings": {

"properties": {

"age": {"type": "integer"},

"email":{"type": "keyword"},

"name":{"type": "text"}

}

}

}

* 添加新的字段映射

PUT /my\_index/\_mapping

{

"properties": {

"employee-id": {

"type": "keyword",

"index": false

}

}

}

* 更新映射

对于已经存在的映射字段，我们不能更新，更新必须创建新的索引进行数据迁移。

* 数据迁移

先创建出 new\_twitter 的正确映射。然后使用如下方式进行数据迁移

修改bank下的mapping

1. 创建新的索引，指定mapping规则

PUT /newbank

{

"mappings": {

"properties": {

"account\_number": {

"type": "long"

},

"address": {

"type": "text"

},

"age": {

"type": "integer"

},

"balance": {

"type": "long"

},

"city": {

"type": "keyword"

},

"email": {

"type": "keyword"

},

"employer": {

"type": "keyword"

},

"firstname": {

"type": "text"

},

"gender": {

"type": "text"

},

"lastname": {

"type": "text",

"fields": {

"keyword": {

"type": "keyword",

"ignore\_above": 256

}

}

},

"state": {

"type": "keyword"

}

}

}

}

结果

{

"acknowledged" : true,

"shards\_acknowledged" : true,

"index" : "newbank"

}

1. 数据迁移

POST \_reindex

{

"source": {

"index": "bank",

"type": "account"

},

"dest": {

"index": "newbank"

}

}

**4、分词**

一个tokenizer(分词器)接收一个字符流，将之分割为独立的tokens(词元，通常是独立的单词)，然后输出tokens流。

例如，whitespace tokenizer 遇到空白字符是分割文本，它会将文本 “Quick brown fox!”分割为[Quick, brown, fox!]

该tokenizer（分词器）还负责记录各个term（词条）的顺序或position位置（用于phrase短语和word proximtiy词近邻查询），以及term所代表的原始word的start和end的character offsets（字符偏移量）（用于高亮显示搜索的内容）

* 标准分词器

POST \_analyze

{

"tokenizer": "standard",

"text": "The 2 QUICK Brown-Foxes jumped over the lazy dog's bone."

}

* 安装自己的分词器（ik）

http://github.com/medcl/elasticsearch-analysis-ik/  
下载对应版本，可以复制下载地址到迅雷下，非常快

进入容器内部

docker exec -it 容器id /bin/bash

* ik zip 解压

unzip elasticsearch-analysis-ik-7.4.2.zip

修改权限  
chmod -R 777 ik/

重启elasticsearch。

* 使用ik  
  ik\_smart

POST \_analyze

{

"tokenizer": "ik\_smart",

"text": "欢迎您的到来"

}

结果

{

"tokens" : [

{

"token" : "欢迎您",

"start\_offset" : 0,

"end\_offset" : 3,

"type" : "CN\_WORD",

"position" : 0

},

{

"token" : "的",

"start\_offset" : 3,

"end\_offset" : 4,

"type" : "CN\_CHAR",

"position" : 1

},

{

"token" : "到来",

"start\_offset" : 4,

"end\_offset" : 6,

"type" : "CN\_WORD",

"position" : 2

}

]

}

ik\_max\_word

POST \_analyze

{

"tokenizer": "ik\_max\_word",

"text": "欢迎您的到来"

}

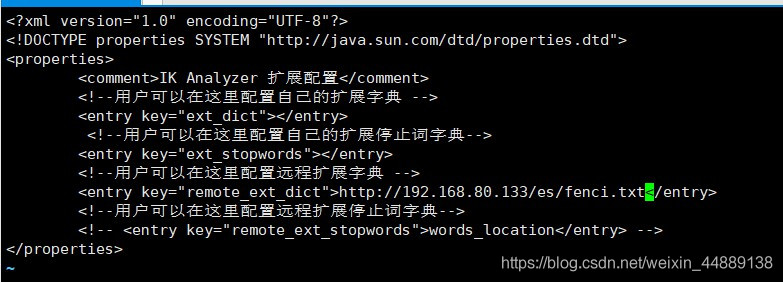
额外  
安装wget,unzip

yum install wget

yum install unzip

* 自定义词库

修改/usr/share/elasticsearch/plugins/ik/config中的IKAnalyzer.cfg.xml  
直接修改外部挂载的文件



**重启容器 -》 如果测试失败 -》看下面第5步最后一点**

**5、安装nginx（为自定义词库创建）**

* 随便启动一个nginx实例，只是为了复制出配置
  + docker run -p 80:80 --name nginx -d nginx:1.10
* 将容器内的配置文件拷贝到当前目录；docker container cp nginx:/etc/nginx . (后面还有个点,且点前面有空格)
* 修改文件名称：mv nginx conf 把这个conf移动到/mydata/nginx下
* 终止原容器：docker stop nginx
* 删除容器 docker rm 容器id
* 创建新的nginx

docker run -p 80:80 --name nginx  
-v /mydata/nginx/html:/usr/share/nginx/html  
-v /mydata/nginx/logs:/var/log/nginx  
-v /mydata/nginx/conf:/etc/nginx  
-d nginx:1.10

* 访问 主机地址

在/mydata/nginx/html中创建index.html

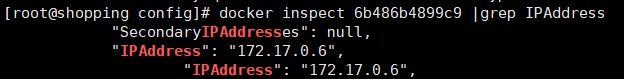
* 创建分词文本

mkdir es  
vi fenci.txt

访问：  
http://192.168.80.133/es/fenci.txt

* 注意问题：我使用的docker需要像之前安装kibana找到ealsticsearch一样，需要得到docker帮它创建的ip,不能直接使用主机ip修改IKAnalyzer.cfg.xml

如图


这样就可以配置成功。

**6、Elasticsearch-Rest-Client**

**1、9300:TCP**

* spring-data-elasticsearch:transport-api.jar
  + spirngboot版本不同
  + 7.x已经不建议使用；8后就要废弃

**2、9200:HTTP**

* JestClient：非官方；更新慢
* RestTemplate：模拟发送HTTP请求，ES很多操作需要自己分装，麻烦
* HttpClient：同上
* Elasticsearch-Rest-Client：官方RestClient，封装了ES操作，API层次分明，上手简单