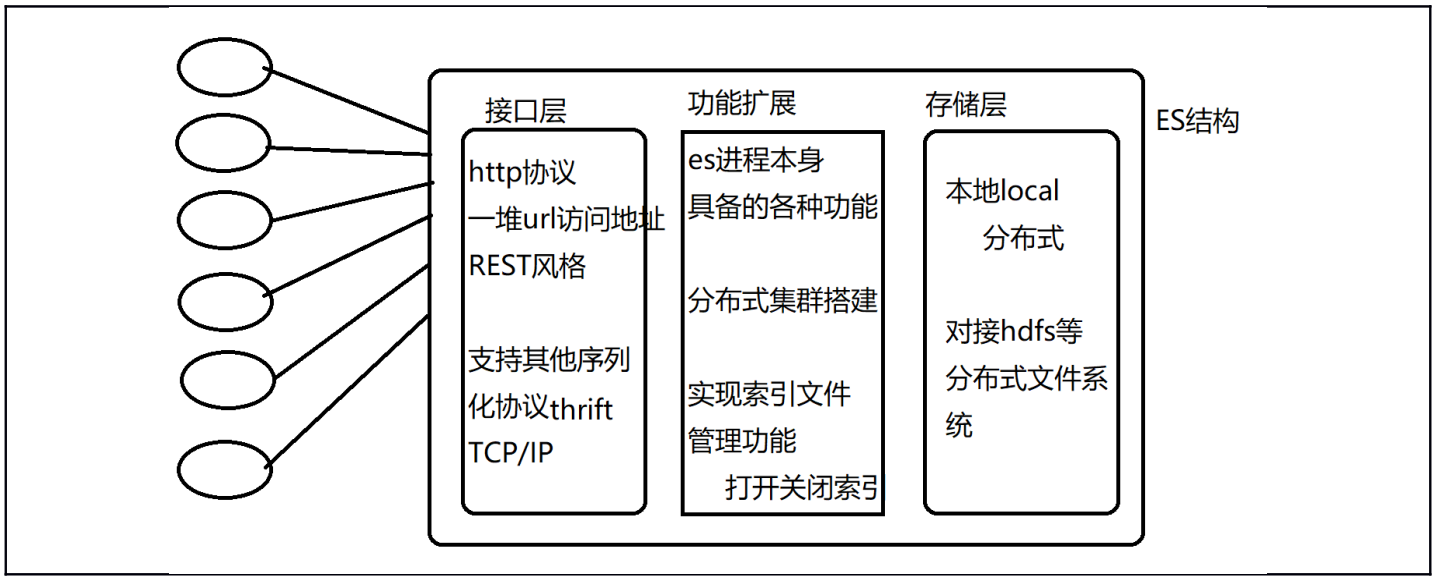
一 ES 简介

Shay Banon 在 2004 年创造了 Elasticsearch 的前身，称为 Compass。在考虑 Compass 的第三个版本时，他意识到有必要重写 Compass 的大部分内容，以“创建一个可扩展的搜索解决方案”。因此，他创建了“一个从头构建的分布式解决方案”，并使用了一个公共接口，即 HTTP 上的 JSON，它也适用于 Java 以外的编程语言。Shay Banon 在 2010 年 2 月发布了 Elasticsearch 的第一个版本。

ES 是一个基于 lucene 实现的搜索服务.当客户端调用 es 的功能时,可以实现全文检索的各种操作,并且不需要关心 lucene 的底层代码.只要客户端支持 http 协议就可以了.

二 ES 结构



1 存储层

可以利用 lucene 创建索引时，决定将索引存储在哪里（都是分布式的）
本地:数据分片切分
远程 hdfs:分布式文件系统

2 功能扩展

es 做一个启动的服务,可以创建集群,实现高可用分布式
扩展了对 lucene 创建的索引管理功能

3 接口层

外界客户端可以通过哪种途径连接 es 操作 es

命令操作:http 协议
java 代码:TCP/IP 协议

三 REST 风格

1 12000 年 http 创始人一篇论文

论文主旨:
大家正在使用的 http 协议方式,和我创建 http 协议的目的不一样---你们在乱用,瞎用,不会用
遵循 REST 风格---你要不会用,就了解什么是 REST 风格

2 URI 的使用:

表示资源 ID，不同的 URI 地址应该指向不同的资源，单独看这个概念很多系统 http 请求接口设计就已经违反了 URL 定义。
例如：对同一商品使用了不同的 URI 进行操作
新增商品: product/save
修改商品: product/update
查询商品: product/query
满足 URI:
product/manage/{productId},新增商品,修改商品,查询商品访问一个地址

3 什么是 REST 风格

representational state transfer(表象性状态转变).
http 请求方式的使用
REST 风格中定义
put 请求表示新增
post 请求表示整体修改覆盖

delete 请求表示删除

get 请求表示查询

配合 uri 表示资源的定义,可以使用不同的请求方式达到增删查改的操作目录

新增商品

put:product/manage/{productId}

修改商品

post:product/manage/{productId}

删除商品

delete:product/manage/{productId}

查询商品

get:product/manage/{productId}

REST 风格

uri 表示访问的资源是谁;

http 请求方式表示要对资源做什么操作;

4 为什么要满足 REST 风格

系统的版本上下很容做成兼容

不满足 REST 风格,版本兼容就要花费更多的成本

例如:

版本 1.0

post:product/manage/save 进行新增

版本 2.0

post:product/manage/insert 进行新增

很多 web 应用框架都支持 REST 风格,比如 springmvc

@RequestMapping(value="product/mange/save",method=RequestMethod.GET)

方法 1:处理 get 请求

@RequestMapping(value="product/mange/save",method=RequestMethod.POST)

方法 2:处理 post 请求

四　ES 安装启动配置

1　目录结构

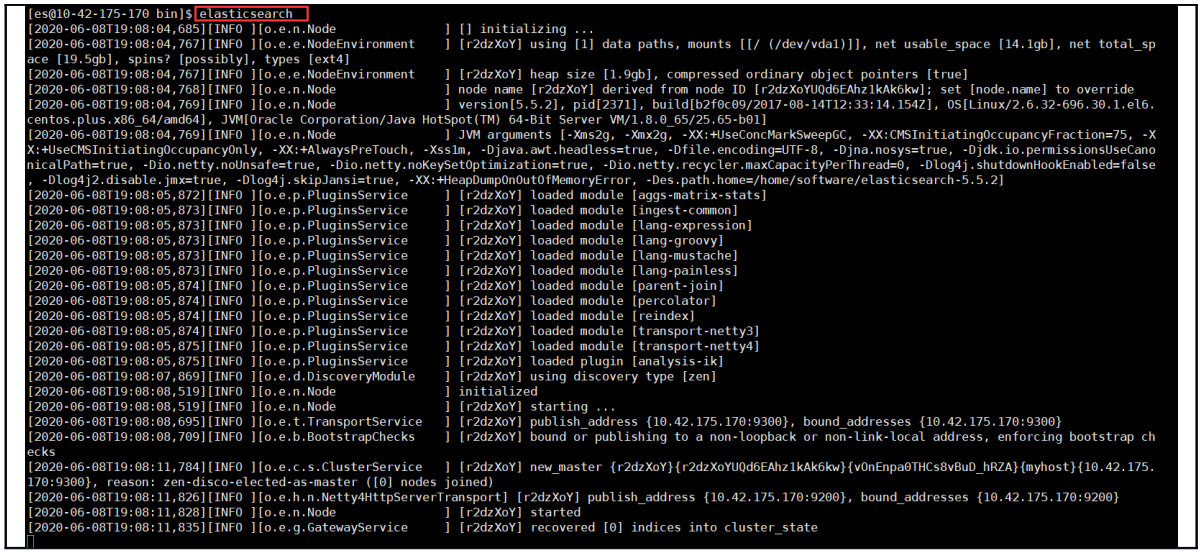
bin:启动命令.
config:
　　elastcisearch.yml:es 节点核心属性配置文件.
　　jvm.options:es 启动的 jvm 配置 占用内存.
logs:日志文件.
data:一个 es 节点保存的所有索引,集群状态数据(存在的话会影响集群搭建)
plugins:IK 分词器插件

2　ES 启动

不允许 root 用户启动软件.使用一个有权限操作 es 的普通用户 es:123/es:es
启动 logs 无权限报错：　Permission denied
root 启动会导致 logs 文件中出现 root 创建的日志,es 用户无法访问无法使用.删除这个 log 重新启动 es

#su es 从 root 进入 es 用户
$exit 从 es 退出到 root

启动命令
es 根目录 bin 文件夹下:
#elasticsearch 在控制台输出日志
#elasticsearch -d 在后台运行



3 es 的配置文件

在 ES 根目录 config 文件夹下，编辑配置文件 elasticsearch.yml

- 17 行:

cluster.name

自定义的集群名称,默认 elasticsearch. 同一个集群所有节点,名字一致

- 23 行:

node.name

节点名称,同一个集群节点名字相互不一样

- 56 行/60 行

network.host

当前服务器被外部访问的 ip 地址可以是域名，默认 127.0.0.1 只允许本机访问这个 es 节点

http.port

http 协议访问操作 es 时需要通过这个端口.默认 9200

- 69 行

discovery.zen.ping.unicast.hosts

集群发现节点的协调器 list,保存了集群中所有协调器的 ip 地址--搭建集群使用

- 73 行

discovery.zen.minimum_master_nodes

防止集群脑裂的配置，默认已被注释的

五　ES 索引分片结构

1　http 协议创建索引观察结构

- 调用创建索引的命令

curl 支持在 linux 系统发送 http 请求

-X 请求方式 默认 GET

curl -XGET url

curl -XPOST url

....

-d 添加请求体数据

curl -XPOST -d '请求体数据' url

创建一个索引 index01：没有数据，是空的

```
[root@10-42-175-170 ~]# curl -XPUT http://10.42.175.170:9200/index01
{"acknowledged":true,"shards_acknowledged":true}
```

2　索引分片结构

上例中创建完 index01 索引后会创建以下目录，建了 5 个索引文件

分布式中可以把 5 个索引文件中的某几个放到其它节点。

1DiLOGd3ReCgZi5X9jZifQ 索引 index01 的 id

```
[root@10-42-175-170 1DiLOGd3ReCgZi5X9jZifQ]# pwd
/home/software/elasticsearch-5.5.2/data/nodes/0/indices/1DiLOGd3ReCgZi5X9jZifQ
[root@10-42-175-170 1DiLOGd3ReCgZi5X9jZifQ]# ll
total 24
drwxrwxr-x 5 es es 4096 Jun  8 19:10 0
drwxrwxr-x 5 es es 4096 Jun  8 19:10 1
drwxrwxr-x 5 es es 4096 Jun  8 19:10 2
drwxrwxr-x 5 es es 4096 Jun  8 19:10 3
drwxrwxr-x 5 es es 4096 Jun  8 19:10 4
drwxrwxr-x 2 es es 4096 Jun  8 19:10 _state
[root@10-42-175-170 1DiLOGd3ReCgZi5X9jZifQ]#
```

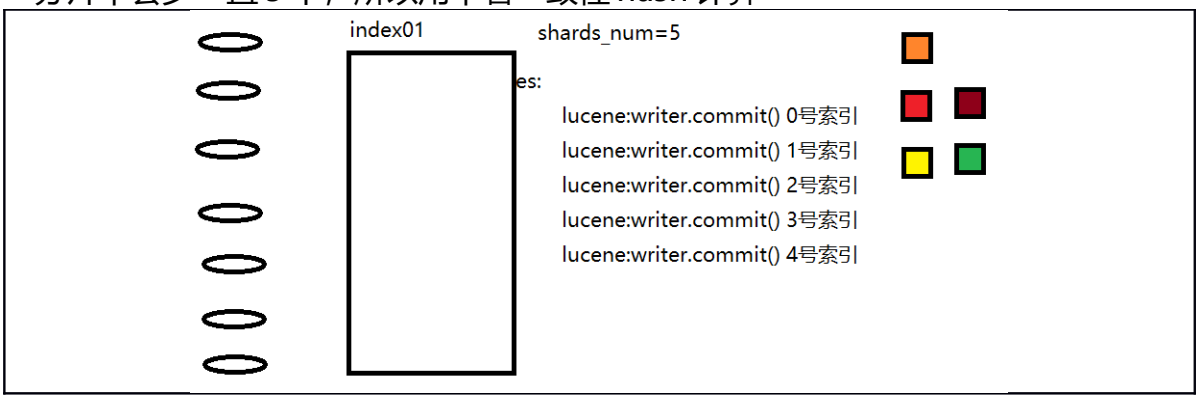
es 默认对每一个索引文件都计算分片,5 个分片.每一个数据分片都是一个完整的 lucene 创建的索引。

```
[root@10-42-175-170 1DiLOGd3ReCgZi5X9jZifQ]# cd 0
[root@10-42-175-170 0]# ll
total 12
drwxrwxr-x 2 es es 4096 Jun  8 19:10 index
drwxrwxr-x 2 es es 4096 Jun  8 19:10 _state
drwxrwxr-x 2 es es 4096 Jun  8 19:10 translog
[root@10-42-175-170 0]# cd ..
[root@10-42-175-170 1DiLOGd3ReCgZi5X9jZifQ]# cd 1
[root@10-42-175-170 1]# ll
total 12
drwxrwxr-x 2 es es 4096 Jun  8 19:10 index
drwxrwxr-x 2 es es 4096 Jun  8 19:10 _state
drwxrwxr-x 2 es es 4096 Jun  8 19:10 translog
```

3 数据的分片计算逻辑

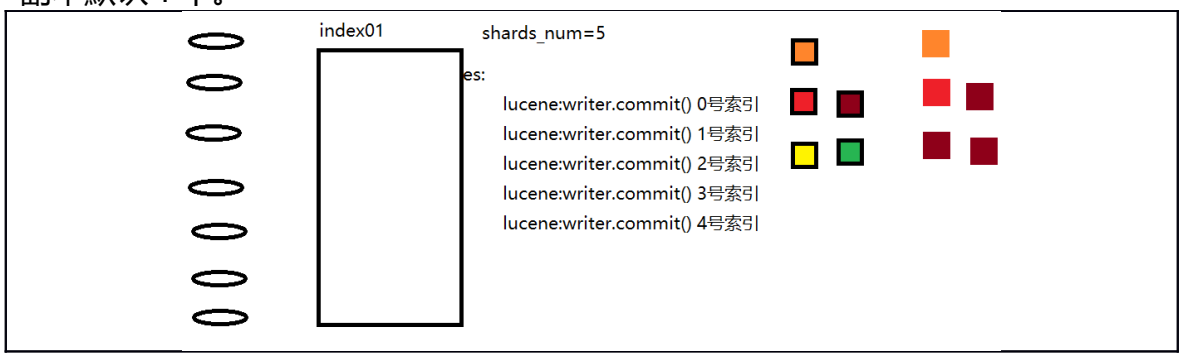
如果我们向 index01 写入数据 放进去一批 document。会根据 document 的 id 值进行 hash 取 对 5 取余，每一个 document 都会对应 0-4 号分片其中的一个，最后某个 document 放到一个分片中。

分片不会少一直 5 个，所以用不着一致性 hash 计算



4 分片高可用

es 除了计算默认 5 个分片,默认为每一个分片在其它节点创建一个副本分片(主从复制) 副本默认 1 个。



因为现在只有一个 es 节点,即使计算除了从分片,没地方放.

5 es 的 head 插件使用（查看数据）

/home/presoftware/elasticsearch-head-master 文件夹.

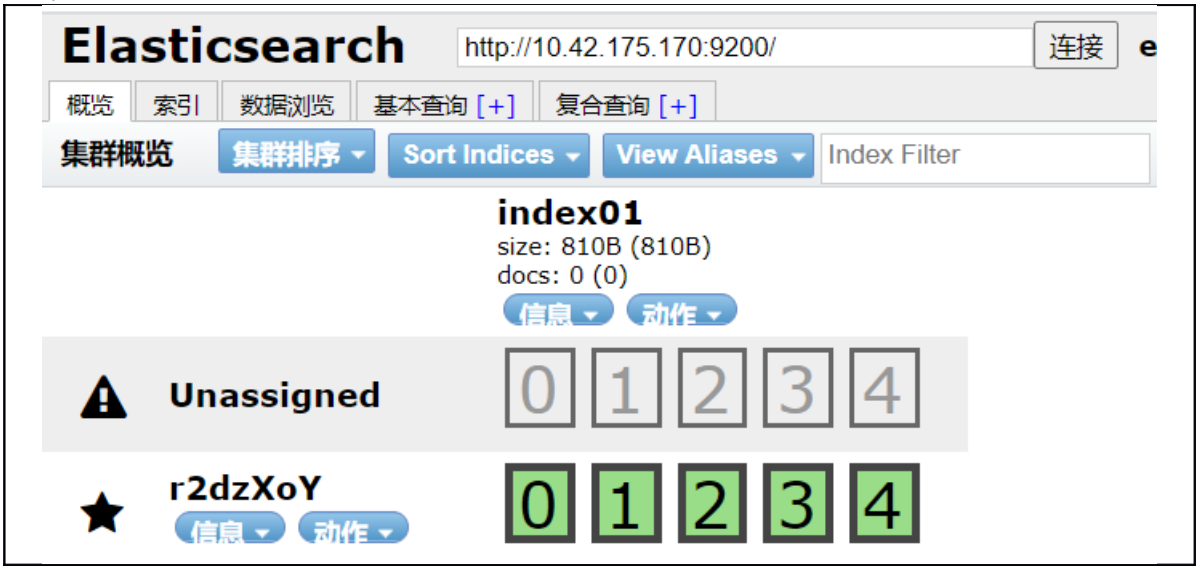
保证当前运行这个 head 插件的服务器的 hosts 文件中有正确的 myhost 映射 ip

在 head 插件的根目录运行启动

#grunt server ： myshost 为系统本地域名，对应 127.0.0.1，外部访问用外网 ip

```
[root@10-42-175-170 elasticsearch-head-master]# grunt server
Running "connect:server" (connect) task
Waiting forever...
Started connect web server on http://myhost:9100
```

网页访问：



六　ES http 协议命令

通过 REST 风格的 http 请求,实现对 ES 的各种操作.

1　索引管理

1.A　创建索引

新增一个 index02 的索引

```
[root@10-42-175-170 ~]# curl -XPUT http://10.42.175.170:9200/index02
{"acknowledged":true,"shards_acknowledged":true}
```

1.B 索引读限制

es 在创建索引的底层使用 lucene 逻辑,但分配更多线程资源管理索引，例如对一个索引的读的限制.可以在访问一个索引时携带一个参数等。

{"blocks.read":true/false}： true 表示开启读限制,false 不开启读限制.

```
[root@10-42-175-170 ~]# curl -XPUT -d '{"blocks.read":true}' http://10.42.175.170:9200/index01/_settings
{"acknowledged":true}
```

读取时卡主：表示限制读取了

```
[root@10-42-175-170 ~]# curl -XGET http://10.9.48.69:9200/index01
```

1.C查询索引

查询 index01 索引

```
[root@10-42-175-170 ~]# curl -XGET http://10.42.175.170:9200/index01
```


```
{
    "index01": {
        "aliases": {},
        "mappings": {},
        "settings": {
            "index": {
                "creation_date": "1591617545339",
                "number_of_shards": "5",
                "number_of_replicas": "1",
                "uuid": "17azJw5iSg6yQdSZ3WU-Uw",
                "version": {
                    "created": "5050299"
                },
                "provided_name": "index01"
            }
        }
    }
}
```


还可以查询多个索引

```
curl -XGET http://10.9.48.69:9200/index01,index02
```

1.D 删除索引

```
curl -XDELETE http://10.9.48.69:9200/index02
```

1.E打开/关闭索引

关闭

```
#curl -XPOST http://10.9.48.69:9200/index01/_close
```

打开

```
#curl -XPOST http://10.9.48.69:9200/index01/_open
```

一旦打开,操作正常执行,可以访问 index 所有数据,状态,document 搜索
一旦关闭,在当前 es 里就再也不能执行与这个所有有关的任何命令.

一些陈旧历史数据,不能删除数据,可以减少系统对他管理的线程资源分片,关闭

2 文档管理

操作完索引后,可以通过 es 的命令,和接口创建删除等管理索引文件,但是任何一个索引文件中都需要保存大量的 document 文档;客户端连接 es 的服务器,传递给 es 一批文档数据,es 才会按照要去写入到索引中.

2.A 新增文档

格式：curl –XPUT –d 文档内容 json url

```
curl -XPUT -d '{"id":10,"title":"es 简介","content":"es 好用好用真好用"}'
http://10.9.48.69:9200/index01/article/1
```

index01：索引名

article：type

1：文档 id

type 类型是什么? es 中维护管理的数据整体结构,完全可以对照 mysql 理解

mysql	ES
database	index
tables	type:同一个类型的 document 域属性结构一致的
rows	document
columns	fields

返回结果：

```
{
  "_index": "index02",
  "_type": "article",
  "_id": "1",
  "_version": 1,对同一个文档的写操作增删改都会在版本 version 进行自增 1,目
```

```
的是分布式高可用中保持数据复制的有效判断.数据一致性。
   "result": "created",
   "_shards": {
      "total": 2,
      "successful": 1,
      "failed": 0
   },
   "created": true
}
```

2.B 获取文档

明确你获取文档的 3 个坐标属性 index type id

2.B.a 单个文档

```
curl -XGET http://10.9.48.69:9200/index02/article/1
{
   "_index": "index02",
   "_type": "article",
   "_id": "1",
   "_version": 1,
   "found": true,
   "_source": {
      "id": 10,
      "title": "es 简介",
      "content": "es 好用好用真好用"
   }
}
```

2.B.b 获取多个文档

需要在请求体中,以参数形式,携带多个文档对应的 index/type/id

```
格式：Curl –XGET json 数组 url
Curl -XGET -d '{"docs":[{"_index":"index02","_type":"article","_id":"1"},
{"_index":"index02","_type":"article","_id":"2"}]}' http://10.9.48.69:9200/_mget
```

2.C删除文档

为了在集群中对主从分片复制有一致性判断,version 在删除时也会自增 1

```
curl -XDELETE http://10.9.48.69:9200/index01/article/1
```

3 搜索管理

3.A match_all

查询:

```
query 结构
{
  "query": {
    "match_all": {

    }
  }
}
curl -XGET http://10.42.175.170:9200/index01/_search '{"query": {"match_all": {}}}'
```

返回结果：查询返回结果格式基本相同

```
{
    "took": 8,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "failed": 0
    },
    "hits": {包含的数据就是通过前查询方式计算的,读到的所有内容
        "total": 1, topDocs.getTotalHits
        "max_score": 1.0,
        "hits": [{数组,相当于 topDocs 属性 scoreDocs ScoreDoc[]
            "_index": "index01",三个坐标
            "_type": "product",
            "_id": "123456",
            "_score": 1.0,匹配分数
            "_source": {source 的值就是新增 -d 携带的字符串
                "productId": "123456",
                "productName": "华为手机",
            }
        }]
    }
}
```

3.B term_query

词项查询,最基本的查询方式之一.可以使用 term_query 验证一下当前数据写入索引时使用的分词器---默认 Standard

```
{
```

```
    "query": {
        "term": {
            "title": "简介"
        }
    }
}
```

```
curl -XGET http:// 10.42.175.170:9200/index01/_search -d '{"query":{"term":{"title":"简介"}}}'
```

默认如果是 standard,不满足我们定义数据分词计算需求,就要使用中文分词器,怎么指定.

3.Cmatch_query

可以不关心分词器，只提供查询的文本，es 会使用分词器对文本先解析，然后对每个词项用 termquery 挨个查询。

```
{
    "query": {
        "match": {
            "title": "大数据简介"
        }
    }
}
```

```
curl -XGET http:// 10.42.175.170:9200/index01/_search -d '{"query":{"match":{"title":"大数据简介"}}}'
```

title 创建多个 TermQuery
首先对 title 进行分词，默认 Standard。大数据简介-->分词器分词-->大 数 据 简 介
5 个 termQuery 挨个查,所有结果返回

七 IK 分词器和 Mapping 设置

1 IK 分词器插件

1.A 插件安装

es 中没有 ik 分词器支持,无法指定这个分词器的使用.所以先来安装插件.
所有分词器代码都会按照 es 能够加载的插件封装.将插件解压到对应 es 根目录 plugins 中.
ik 插件包 /home/resources/elasticsearch-analysis-ik-5.5.2.zip

```
[root@10-9-48-69 resources]# unzip /home/resources/elasticsearch-analysis-ik-5.5.2.zip -d /home/software/elasticsearch-5.5.2/plugins/
```

1.B 插件测试

重启 es ： 让 es 重新加载 plugins 下的内容

发送一个请求 使用一个已有的索引,计算分词器的效果

```
http://10.42.175.170:9200/index02/_analyze?analyzer=ik_max_word&text=中华人民共和国
```

参数

analyzer：分词器名字 ik_max_word

text：计算测试的文本

2 Mapping 映射

当 es 安装完了 ik 分词器之后,并不会在创建索引 document 数据时自动使用 ik 分词器。

创建数据之前配置一个叫做 mapping--映射，决定数据 document 结构（可以理解位数据库的表结构）。

类似我们之前用 lucene 创建时,指定域属性。

2.A Mapping 是什么

是每个索引中的一个属性值(索引管理中查询索引时可以看到)，定义了当前索引中某个类型的具体结构。

联想数据库的表格结构(schema),不同的是：table 的结构是在创建表格之前就定义好的。

mapping 有 2 种类型.

动态 mapping

静态 mapping

2.B 动态 Mapping

对于一个索引中的某个类型中的某一批 document,可以通过在添加数据时自动生成 mapping 的映射关系（域属性类型，计算分词分词器等）

新增索引时不会创建 mapping，第一次添加 document 时会生成

比如:

新增数据时,{"name":"wanglaoshi"},对应动态 mapping 把数据按照字符串自动生成

下面是查询索引返回的一部分数据:

```
curl -XGET http://10.42.175.170:9200/index01
```

```
        "mappings": {
            "product": {
                "properties": {
                    "name": {
                        "type": "text",
                        "fields": {
                            "keyword": {
                                "type": "keyword",
                                "ignore_above": 256
                            }
                        }
                    }
                }
            }
        }
```

json 是字符串时,默认 type:text 底层就是 lucene 的 TextField

json 是个数字数据,默认 type:long 底层 lucene 创建 document 使用 LongPoint+StringField

json 是个时间字符串, 默认 type:date 底层都是 lucene 的 StringField

type: 生成 type 类型值 "text" 表示这个 name 域属性是个字符串

fields:是额外的指定

type:"keyword" 当前有 fields 属性的 name 除了 text 特性

还有 keyword 底层就是 StringField 以完整的内容保存 name 字符串

ignore_above:256 但是当 name 字符串值长度超过 256 字节时,不会在保存完整字符串.

总结动态 mapping:

mapping 索引中定义类型里 document 结构的一个属性
动态 mapping:什么时候写数据,就会在什么时候自动生成对应的动态结构

字符串: type:text 表示底层会按照分词计算,还有附加属性 type:keyword 并且在不大于 256 字节长度时保存完整字段值

整数数字类型 type:long
浮点数:type:float
日期类型:type:date
地理位置类型....

动态 mapping 中默认指定了一个属性，标准分词器分词
analyzer:standard
如果动态 mapping 满足所有数据的创建需求,则可以继续使用,不满足可以换成静态 mapping(自定义数据类型结构)

2.C静态 Mapping

手动定义 mapping：创建索引时,人为定义 mapping 结构。这样就不会再生成动态的 mapping。.

新建一个 index06 将已经编辑好的 mapping 放进去作为请求体数据.

analyzer：指定分词器

```
{
    "mappings": {
        "book": {
            "properties": {
                "content": {
                    "type": "text",
                    "analyzer": "ik_max_word"
                },
                "id": {
                    "type": "integer"
                },
                "title": {
                    "type": "text",
                    "analyzer": "ik_max_word"
                }
            }
        }
    }
}
```

```
curl -XPUT -d '{"mappings":{"book":{"properties":{"content":
{"type":"text","analyzer":"ik_max_word"},"id":
{"type":"integer"},"title":
{"type":"text","analyzer":"ik_max_word"}}}}}' http://
10.42.175.170:9200/index06
```

验证 mapping 有没有设置成功

- 写入一个 document 携带 content id title
- termQuery 查询 title:简介 查询有结果说明 ik 分词器生效 mapping 生效的.

```
curl -XGET http://10.9.48.69:9200/index06/_search -d '{"query":
{"term":{"title":"简介"}}}'
```

八 ES 集群

1 es 集群特点

es 天生支持集群的配置,即使我们只启动一个 es 节点,也是以集群的逻辑运行的.相对于之前的 redis mycat 集群搭建简单多了。

2 es 集群结构和角色

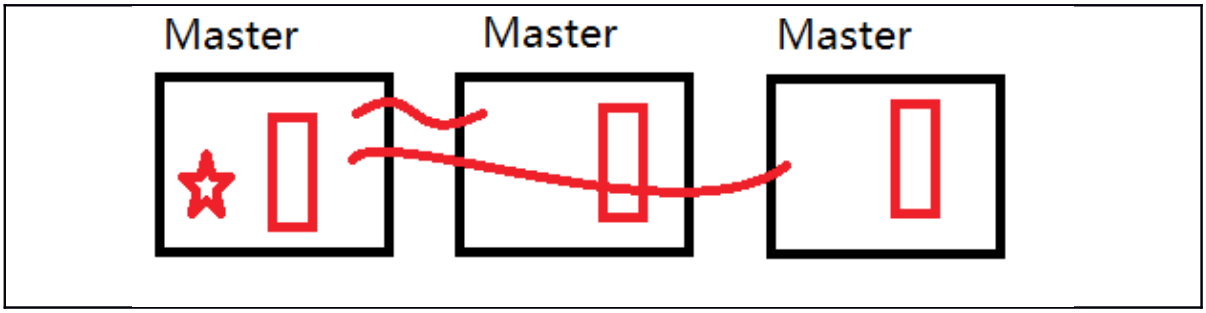
2.A 主节点（master）

功能:是集群的一个大脑。主要功能是计算管理一批集群元数据 。

元数据:

meta-data,描述数据的数据

master 中对 es 集群管理的元数据包括：分片的个数，分片的名字，分片的存储位置,从分片个数，从分片的存储位，,document 文档元数据(index,type,id,version) 包括集群状态等。



Master 节点需要在集群中指定多个,同一时间只能选举其中一个作为 leader，其他 master 节点同步元数据,一旦 leader 宕机完蛋,剩余 master 选举顶替—保证元数据高可用

元数据生成:

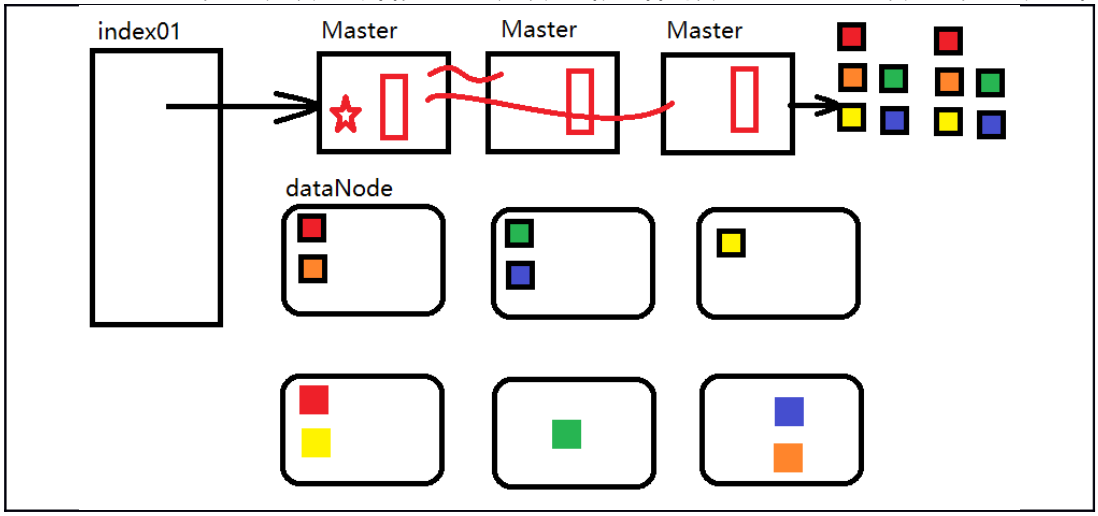
集群中的基础数据（状态,节点信息 ）在创建集群时,元数据文件就生成了

写入索引:master 节点 leader 开始计算(分片个数,从分片,分布式计算等等)

2.B 数据节点（data）

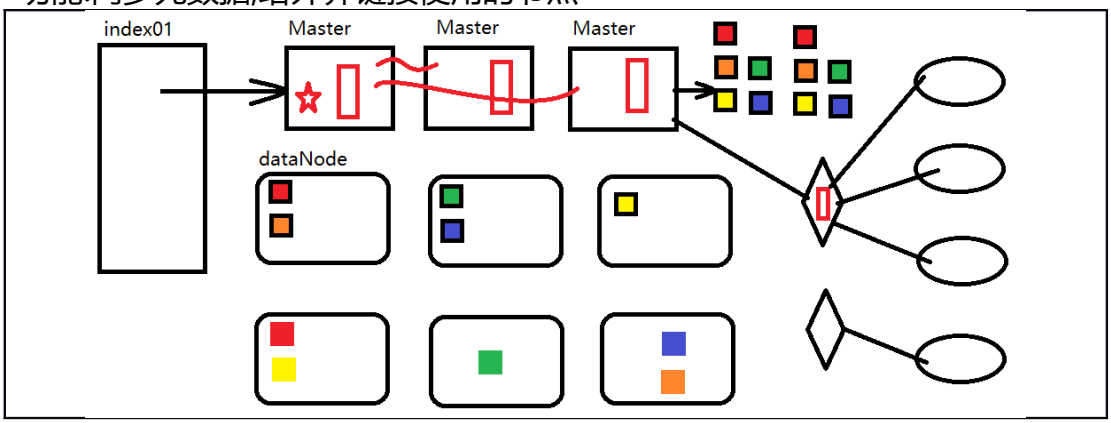
- 功能:读写索引文件

当 master 计算完元数据库,更新完元数据,开始对数据节点进行数据读写功能;



2.C负载均衡节点

功能:同步元数据,给外界链接使用的节点

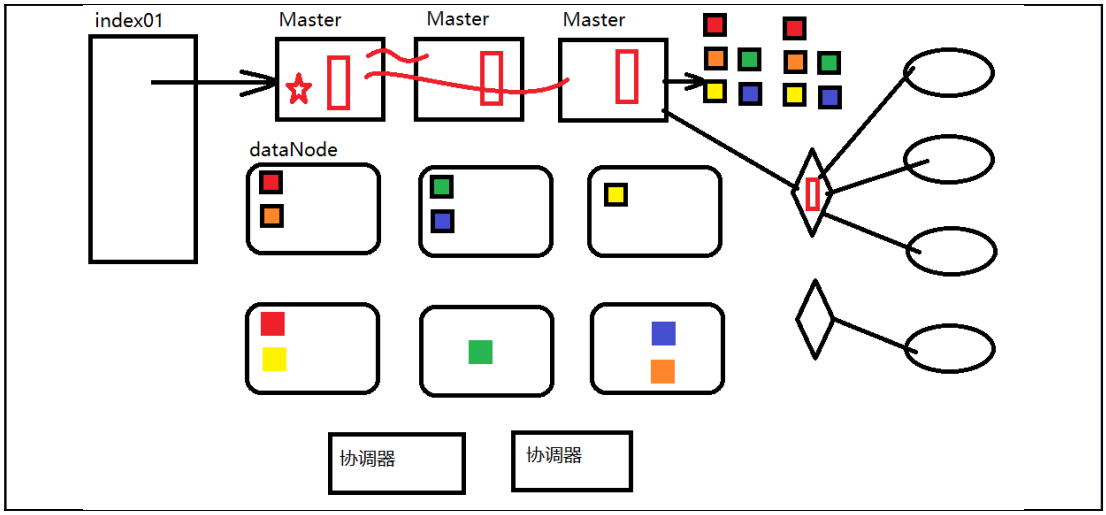


2.D 协调器

- 功能:链接集群的枢纽

协调器可以通过链接一个集群的所有节点,将集群链接成一整个服务.

进群工作中,所有角色节点都要链接协调器才能完成集群的后续逻辑



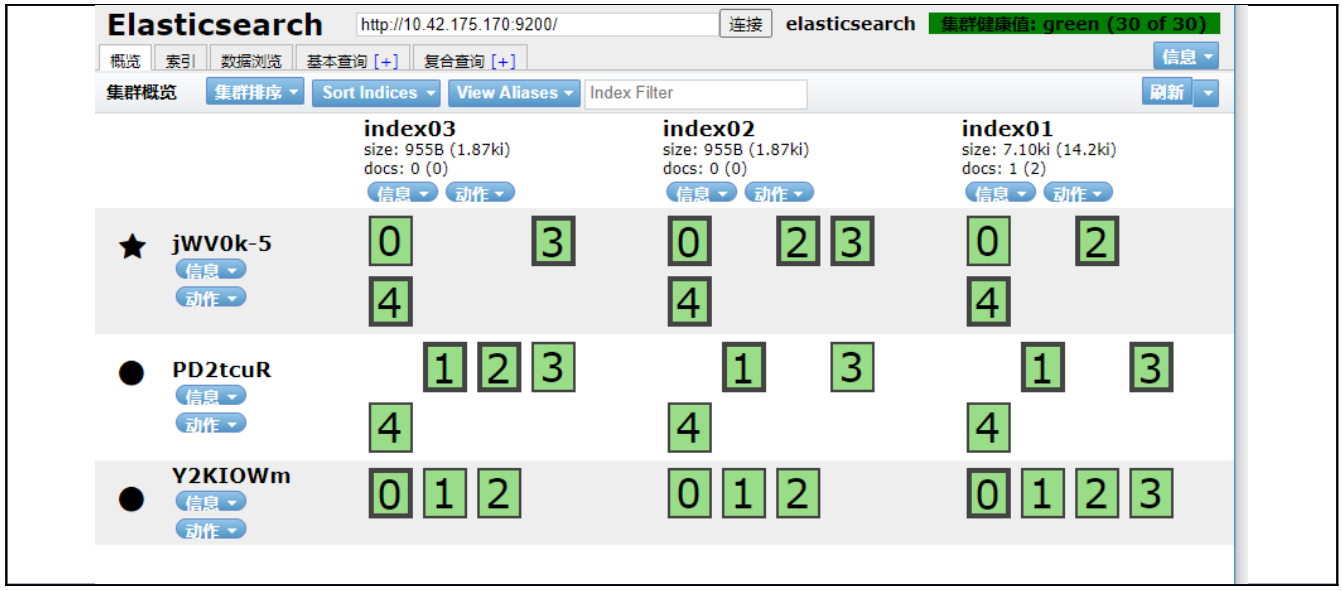
3 启动创建集群

在单个节点配置基础之上完善集群的创建

- 准备 3 个节点每个主机运行一个 es 进程
    - 删除 es 根目录的 data 文件夹.不删除可能会有数据冲突
    - 保证 3 个节点的 plugins 环境是一致（例如 IK 分词器插件）
- 配置文件修改 elasticsearch.yml
    - 确保同一个集群加到一起 即每个 cluster.name 值一致
    - network.host（es 被外部访问的地址）根据主机定义
    - 协调器的 list:在 3 个 es 中选择至少 1 个作为协调器运行,保证协调器高可用则至少 2 个

      discovery.zen.ping.unicast.hosts: ["es1 的地址","es2 的地址"]
    - 防止脑裂发生的配置（73 行） 最小有效 master 个数 2(3 个 master 过半)

      discovery.zen.minimum_master_nodes: 2
- 挨个启动每一个 es 节点
- 效果查看：三个主机，新增三个索引 index01/2/3，都进行了数据分片
    - 五角星的位 leader



九　ES 集群原理—脑裂

在 es 集群中,由 leader 管理元数据,并且其他 master 同步备份的元数据，这种结构容易出现脑裂

1　什么是脑裂

es 脑裂是集群执行选举时发生的，即一个集群出现多个 leader。问题严重在于多个 leader 保存多个元数据且不一致。其它 master 只会从其中一个 leader 同步元数据。.

2　脑裂的原因

es 的 master 的 leader 是通过选举产生的，一旦发现现役 leader 宕机,则从其他 master 里选出新的 leader.有一种情况会在现役 master 没有宕机时,就从其他 master 选择新的 leader,例如：网络波动（误以为宕机）

三台 master 设备 ABC 在互相通信的时候可能会出现网络波动导致分区为 A|BC，即会被误判断为 A down 机，A 也认为 BC 连不上了。这个时候 A 会自己对自己进行选举，BC 也会进行选举，如果没有设置脑裂值为大于等于集群节点的半数，可能会出现一个集群有两个 master 作为 leader 出现，又由于 master 之间只会从一个 leader 的元数据进行数据同步保证数据的一致性，现在有两个 leader 会出现两个 leader 各自保存的数据不一致最后整个集群的数据将会混乱

3 脑裂的防止

配置文件中 配置了一个最小有效 master 个数:集群中所有 master 节点个数过半数.在不满足最小 master 个数的选举中,leader 是无效的.当这个值生效时.

集群无论如何选举,总会至多一个有效 master 存在

例如：在脑裂的参数原因的例子中，如果设置了有效 master 为过半数 2，当 A 被误以为 BC 宕机，此时 A 不会再自我选举，因为只有 1 个节点不满足过半数 2。

十 ES 集群原理—选举机制

链接协调器执行选举逻辑,分为 4 步.

第一步:	启动一个 master 之后立刻链接协调器，获取集群的所有 master 节点信息，信息中 activeMaster 值保存着当前集群的 leader 节点，进入第二步
第二步:	判断 activeMaster 是否为空。若集群没有选举 leader 则 activeMaster 为空。若集群已经选举则 activeMaster 不为空。为空进入第三步，不为空当前节点选举结束(所有节点都会结束在这一步)
第三步:	将当前集群中所有 master 节点信息,放到一个后备 master 集合中（candidateList），进入第四步
第四步:	判断 candidateList 中的 size 是否大于等于最小 master 数量（防止脑裂的哪个设置？）.如果满足将会把 candidateList 中的节点进行对比，id 值大的会暂定为 activeMaster 重新回到第一步. 如果不满足，直接重新回到第一步

如果集群 master 节点宕机

master 会不断链接协调器.

leader 宕机:

协调器发现,立刻将 activeMaster 清空,剩余 2 个节点发现 activeMaster 为空,重新执行选举逻辑,直到从剩余两个中选择一个

备用 master 宕机:

协调器发现,其他 master 就发现了.

宕机一个,candidateList 少了一个,剩余 2 个,满足最小 master 数量

宕机 2 个,candidateList 少了 2 个,剩余 1 个,不满足最小 master.

十一　JAVA 客户端 TransportClient

Transportclient 对象,底层需要 java 的 Socket 支持,创建连接时需要通过 Socket 代码获取 ip 地址和端口.

```
public void init() throws UnknownHostException {
    //给 client 赋值
    //传递一个 setting 值，定义连接的集群名称 elasticsearch
    //Settings settings = Settings.builder().put("cluster.name",
"elasticsearch").build();
    //默认集群名称就是 elasticsearch，锁不用手动设置了
    transportClient=new PreBuiltTransportClient(Settings.EMPTY);
//创建一个节点的 ip 端口
    InetSocketTransportAddress address=
            new
InetSocketTransportAddress(InetAddress.getByName("10.42.175.170"),9300);
    transportClient.addTransportAddress(address);
}
```

1　引入依赖

```
<!--#####搜索 elasticsearch##############-->
        <!--elasticsearch-->
        <dependency>
            <groupId>org.elasticsearch.client</groupId>
            <artifactId>transport</artifactId>
            <version>5.5.2</version>
            <exclusions>
                <exclusion>
                    <!--不使用默认的 2.4.6 版本-->
                    <groupId>org.elasticsearch</groupId>
                    <artifactId>elasticsearch</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <!--使用 5.5.2 版本-->
        <dependency>
            <groupId>org.elasticsearch</groupId>
            <artifactId>elasticsearch</artifactId>
            <version>5.5.2</version>
        </dependency>
```

2　索引管理

```
public void indexManage() throws IOException {

    //创建索引 http://10.9.151.60:9200/index01 put
    //获取 admin 可以管理集群 cluster 也可以管理 indices
```

```
    AdminClient admin = transportClient.admin();
    //1、获取操作 index 的管理对象
    IndicesAdminClient indices = admin.indices();
    ClusterAdminClient cluster = admin.cluster();
    //########建一个索引 index07#################
    //indices.create();//方法调用,命令就发送
    //2、获取 request 对象没有发送命令
    CreateIndexRequestBuilder createIndexRequest =indices.prepareCreate("index07");
    //prepare**方法调用没法送命令,但是获取了返回值 request 对象
    //想把代码命令发送出去，则需要用 get()
    //3、发送请求
    CreateIndexResponse createIndexResponse = createIndexRequest.get();
    //response 解析就是创建索引的返回值 {"acknowlaged","shards_acknowlaged"}
    //4、操作返回值
    createIndexResponse.isShardsAcked();
    createIndexResponse.isAcknowledged();

    //其他的 reques 对象，可以做对应的相关操作
    GetAliasesRequestBuilder GetAliasesRequest = indices.prepareGetAliases("index07");
    GetMappingsRequestBuilder getMappingsRequest =
indices.prepareGetMappings("index02");
    IndicesExistsRequestBuilder indicesExistesRequest =
indices.prepareExists("index07");
    GetMappingsResponse getMappingsResponse = getMappingsRequest.get();
    System.out.println("index07 存在吗:"+indicesExistesRequest.get().isExists());
    //常用的方法,创建索引,不存在时创建索引
    //prepareCreate prepareExists prepareDelete;
}
```

3 文档管理

```
public void docManage() throws JsonProcessingException {
    //1、随便构造一点数据存到 es 中
    Product product=new Product();
    product.setProductCategory("手机");
    product.setProductId("好用");
    product.setProductId("123456");
    product.setProductImgurl("http://image.jt.com");
    product.setProductName("华为手机");
    product.setProductNum(50);
    product.setProductPrice(50.0);
    //http://10.9.151.60:9200/index/article/1 -d
    //2、将对象转化为 json
    ObjectMapper mapper=new ObjectMapper();
    String json = mapper.writeValueAsString(product);
    //3、通过 client 的 prepareIndex 方法获取 request
    //传递坐标的三个参数 索引名，类型，文档 id
    IndexRequestBuilder request = transportClient.prepareIndex("index07", "product",
product.getProductId());
    //4、设置 request 请求数据 json 字符串
    request.setSource(json);
```

```
    //5、发送请求
    IndexResponse indexResponse = request.get();
    //6、处理返回值
    //新建 document 返回字符串 json _index _type _id _version
    long version = indexResponse.getVersion();
    System.out.println("创建 version:"+version);
    //删除 document
    transportClient.prepareDelete("index07","product","123456");
    //获取
    transportClient.prepareGet("index07","product","123456");
}
```

4 搜索功能

```
public void search1() {

    //通过 query 对象的封装 实现不同搜索逻辑
    //搜索的过程的分页时浅查询
    TermQueryBuilder query = QueryBuilders.termQuery("title", "介");
    //搜索
    SearchRequestBuilder request = transportClient.prepareSearch("index02");
    //搜索条件
    request.setQuery(query);
    //搜索分页数据,其实位置,条数
    request.setFrom(0);//limit start
    request.setSize(5);
    SearchResponse searchResponse = request.get();
    //searchResponse 解析查询结果
    SearchHits topDocs = searchResponse.getHits();
    System.out.println("查询总数:" + topDocs.totalHits);
    //遍历数组
    SearchHit[] scoreDocs = topDocs.getHits();
    for (SearchHit hit : scoreDocs) {
        //获取一个 doument 返回结果
        //{"id":"","title"}
        System.out.println("source 的 json:" + hit.getSourceAsString());
    }
}
```

十二　集成 SpringbOOT

1　3.1springboot 自动配置

redis：RedisAutoConfiguration

Elasticsearch: ElasticsearchAutoConfiguration

1.5.9.RELEASE-->2.4.6 版本的 TransportClient

2　手动配置

原理:让框架来管理连接对象.spring 的配置类

让框架管理一个连接 es 集群的 Transportclient 对象

声明一个配置类，spring 加载读取配置文件并创建 bean

```
/**
 * @作者 舒新胜
 * @项目 easymall-2002-all
 * @创建时间 2020/6/9 14:09
 */
@Configuration
//配合@Configuration 实现配置类属性初始化赋值
//和@Value 作用一样可以读取 properties
//自定义前缀，多级赋值
@ConfigurationProperties("easymall.es")
public class ESConfig {

    public List<String> getNodes() {
        return nodes;
    }

    public void setNodes(List<String> nodes) {
        this.nodes = nodes;
    }

    //如果有多个参数，赋值时逗号分隔：easymall.es.nodes=SHU,XIN
    //然后会把分割过后的值交给这个 list
    //###ESeasymall.es.nodes=10.42.175.170:9300

    private List<String> nodes;
    @Bean
    public TransportClient initClient() {
        System.out.println("开始创建 ES 连接客户端");
        System.out.println(Collections.singletonList(nodes));
        PreBuiltTransportClient transportClient = new
PreBuiltTransportClient(Settings.EMPTY);
        for (String node : nodes) {
```

```
            String[] split = node.split(":");
            //ip
            String host=split[0];
            //端口
            int port=Integer.parseInt(split[1]);
            //封装地址
            InetSocketTransportAddress address = null;
            try {
                address = new
InetSocketTransportAddress(InetAddress.getByName(host), port);
            } catch (UnknownHostException e) {
                e.printStackTrace();
                continue;
            }

            transportClient.addTransportAddress(address);
        }
        return transportClient;

    }
}
```

@Configuration：标识为配置类
//配合@Configuration 实现配置类属性初始化赋值
//和@value 作用一样可以读取 properties
//自定义前缀，多级赋值
//加载配置类时，会读取配置文件的 easymall.es.**的值赋值给**变量
//如果变量是另一个配置类，则可以在配置文件中通过 easymall.es.另一个配置类.**给这个配置类中的属性赋值。
@ConfigurationProperties("easymall.es")
@Bean：该注解指定的方法返回值会交给 spring 容器

配置文件设置值，spring 自动赋值

```
###ES
easymall.es.nodes=10.42.175.170:9300,10.9.151.60:9300
```

当使用配置逻辑完成对象的框架管理过程,开发者使用的代码还是最底层客户端代码.一般作为架构师都会对其做 2 次封装,让开发过程变得简单

```
@Component
public class ESUtils {
    @Autowired
    private TransportClient client;
    //一个叫做创建索引,
    public boolean createIndexByName(String indexName){
        return
client.admin().indices().prepareCreate(indexName).get().isAcknowledged();
    }
    //一个叫做判断索引存在
}
```

十三　ELK 家族

1　ELK 介绍

数据采集和整理过程在上述的结构中,没有实现定时执行(手动访问的)
数据源不同,java 代码就要编写很多个.

es 中的数据--索引文件,价值不是体现太好.非常庞大的索引文件,数据量大到一定程度,但是只能用来搜索.

E:elasticsearch,作用就是管理一堆的索引文件
L:logstash,日志数据采集器.作用就是对接数据源(对接很多种数据源),将数据源源源不断的输出到 es 中
Kibana:数据分析插件,作用:链接 es 实现数据的分析.

2　数据采集整理一致性

如果 ELK 使用不用业务层面考虑一致性,logstash 很容易解决,定时执行任务就行;

在数据源发生变动时,一并调用处理缓存和索引的逻辑

十四　CAP 理论

是分布式基础理论.设计一些技术选型的逻辑

C:Consistency 一致性
A:avalibility 可用性
P:partition tolerency/partition 分区容忍度和分区

结论:分布式系统中 CAP 三个方面无法同时满足,只能满足最多 2 个 CA CP AP

P:分区,在分布式环境中,由于网络限制,导致系统之间各个部分相互不可连接,形成分区.分区是一种常态

C:对数据的一致性要求高---强一致性

A:对系统的可用性要求高.

当分区出现时 P,对数据一致性要求高 ---CP 理论

舍弃可用性,对系统加锁,加事务,保证数据修改是一致性

当分区出现时 P,对系统可用性要求高 ---AP

数据一致性要求不高,对系统可用性没有影响

分区不存在网络连接畅通可以 AC

什么时候要求可用性:

微服务调用--可用性

eureka 中的注册信息是一致性体现吗?当一个服务关闭后,保持数据一致性,在 eureka 注册中西立刻剔除

互联网项目系统的普通功能

搜索,查询,登录

什么时候要求一致性:

支付:

客户端数据/服务器钱---第三方支付平台/银行