**ElasticSearch基础上课笔记**

**参考资料：**

<https://www.elastic.co/guide/cn/elasticsearch/guide/cn/index.html>

这个是中文版的，虽然这个是es2.x版本的，但是大家可以发现其核心的技术点并没有太多的变化，仍然具备参考价值。

<https://www.elastic.co/guide/en/elasticsearch/reference/6.0/search-aggregations-metrics-avg-aggregation.html>

6.x英文版api资料

**一．分布式索引介绍**

**1.number\_of\_shards：**分片数量，类似于数据库里面分库分表，一经定义不可更改。主要响应写操作

**2.number\_of\_replicas：**副本数，用于备份分片的，和分片里面的数据保持一致，主要响应读操作，副本越多读取就越快。

**3.分布式索引一定要注意分片数量不能更改，**所以在创建的时候一定要预先估算好数据大小，一般在8CPU16G的机器上一个分片不要超过300g（不怕的话就500g，只要你索引数据结构优化的好，一般的是没问题）。索引会根据分片的配置来均匀的响应用户请求

**4.如果调整了分片数那就要重建索引。**

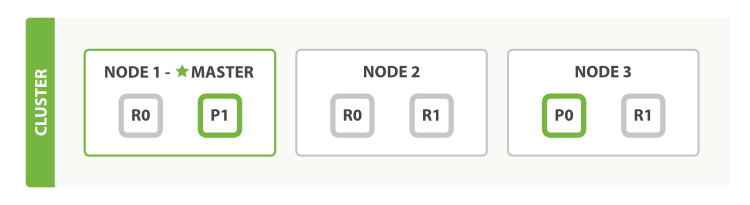
**5.为什么shards不能修改**：

因为我们采取的是分布式索引，假如我们有4篇文档id分别为：1，2，3，4。相信大家再数据库的分表中都知道有一个取模算法。用它来分配记录到对应的表中，其实es也采取的是这种思路。

所以1 3 % 2 ==> 1那么就会在第二个分片上。2 4 % 2 ==> 0那么就会在第一个分片上。

如果这时候你把分片数变了，很显然数据就不对了。所以分片数一旦变化需要重新全部建索引

**6.索引写操作在集群上的操作：**



假设我们集群如上面所示：这上面每个节点都可以接收请求，那么新建的一个流程如下所示：



以下是在主副分片和任何副本分片上面 成功新建，索引和删除文档所需要的步骤顺序：

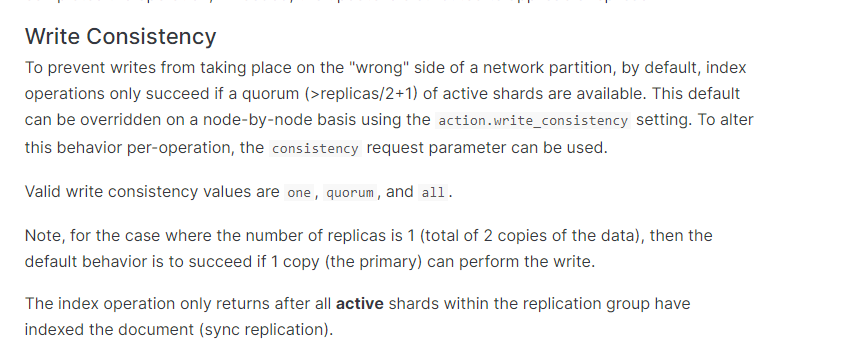
(1)客户端向Node1发送新建、索引或者删除请求。

(2)节点使用文档的\_id确定文档属于分 0。请求会被转发到Node3因为分片 0 的主分片目前被分配在 Node3上。

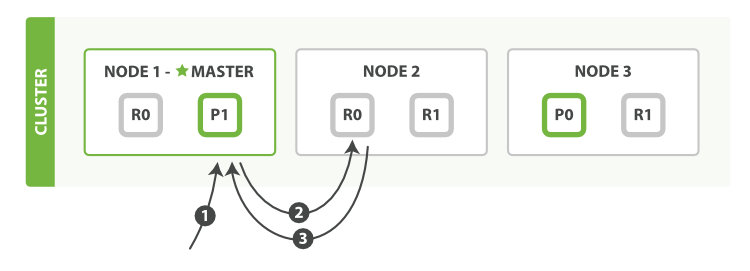
(3)Node 3在主分片上面执行请求。如果成功了，它将请求并行转发到 Node 1和 Node 2 的副本分片上。一旦所有的副本分片都报告成功, Node 3 将向协调节点报告成功，协调节点向客户端报告成功。

在客户端收到成功响应时，文档变更已经在主分片和所有副本分片执行完成，变更是安全的。

（这里课上讲的应该有点问题，现在的es版本是可用shard过半就可以写入，成功后会发送消息给master，我查了es最新的官方文档确实是这样的。所以如果书分片设置为1副本的时候，那默认的就是主分片写如成功了就会返回成功。）以下为es官方文档：



**7.读取文档**



以下是从主分片或者副本分片检索文档的步骤顺序：

(1)客户端向Node1发送获取请求。

(2)节点使用文档的\_id来确定文档属于分片0,分片0的数据在三个节点上都有。 在这种情况下，它会根据负载均衡策略将请求转发到其中一个，比如Node2 。

(3)Node2将文档返回给Node1然后将文档返回给客户端。

在处理读取请求时，协调结点在每次请求的时候都会通过轮询所有的副本分片来达到负载均衡。

在文档被检索时，已经被索引的文档可能已经存在于主分片上但是还没有复制到副本分片。 在这种情况下，副本分片可能会报告文档不存在，但是主分片可能成功返回文档。

**二：Es基础查询语法：**

**中文分词的接入：**

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

安装命令：

./bin/elasticsearch-plugin install https://github.com/medcl/elasticsearch-analysis-ik/releases/download/v6.6.0/elasticsearch-analysis-ik-6.6.0.zip

注意版本号一定要对应

安装完后要重启es

**创建一个索引**

PUT /test

{

"settings" : {

"number\_of\_shards" : 1,

"number\_of\_replicas" : 1

}

更新其replicas状态，但是不能更新shards状态

PUT /test/\_settings

{

"number\_of\_replicas" : 0

}

**创建索引，指定id建立索引**

PUT /test/\_doc/1

{

"name": "赵云",

"age": 30

}

**指定id全量修改索引**

PUT /test/\_doc/1

{

"name": "赵1云"

}

**指定id部分字断修改**

POST /test/\_doc/1/\_update

{

"doc":{

"name":"赵云1"

}

}

**指定\_create防止重复创建,**如果已经存在则失败,以下语句执行第二次会报错

POST /test/\_doc/2/\_create

{

"name":"赵云",

"age":1

}

**使用搜索全部**

GET /test/\_search

**获取指定id**

GET /test/\_doc/1

**不指定id建立索引**，es会默认给你生成一个

POST /test/\_doc/

{

"name": "赵云222",

"age": 30

}

**删除文档:指定id**

delete /test/\_doc/1

**删除索引**

DELETE /test

**结构化创建索引**

PUT /test

{

"settings": {

"number\_of\_shards": 1,

"number\_of\_replicas": 1

},

"mappings": {

"\_doc":{

"properties": {

"name":{"type": "text","analyzer":"ik\_max\_word","search\_analyzer": "ik\_smart"},

"sname":{"type": "text","analyzer":"ik\_smart"},

"enname":{"type":"text","analyzer":"english"},

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

}

}

}

}

Es基础查询

**简单查询：**

（1）主键查询 GET /test/\_doc/1

**(2) 查询所有**

GET /test/\_search

{

"query":{

"match\_all": {}

}

}

**（3）分页查询**

GET /test/\_search

{

"query":{

"match\_all": {}

},

"from":0,

"size":1

}

**复杂查询**

**（1）带条件**

GET /test/\_search

{

"query":{

"match": {"name":"赵云"}

}

}

**（2）带排序**

GET /test/\_search

{

"query":{

"match": {"name":"赵云"}

},

"sort":[

{"age":{"order":"desc"}}

]

}

**(3) 带聚合**

GET /test/\_search

{

"query":{

"match": {"name":"赵云"}

},

"sort":[

{"age":{"order":"desc"}}

],

"aggs":{

"group\_by\_age":{

"terms":{

"field":"age"

}

}

}

}

**analyze api**

GET /test/\_analyze

{

"field": "name",

"text": "my name is zhaoyun and i like eating apples and running"

}

GET /test/\_analyze

{

"field": "sname",

"text": "武汉市长江大桥"

}

从上面的结果中我们可以看出es对齐进行了分词。Es中主要有三种分词：

(1)stander中文的默认分词，这是按照一个个字分分开。此种分词非常占空间会导致倒排索引很大，而且在搜索的时候也会搜出很多不相干的东西。优点就是搜的多。。

(2)English分词器：会提取词干和去掉停用词。假设name不采用英文分词：

PUT /test/\_doc/1

{

"name": "eating apples,

"age": 30

}

然后搜索

GET /test/\_search

{

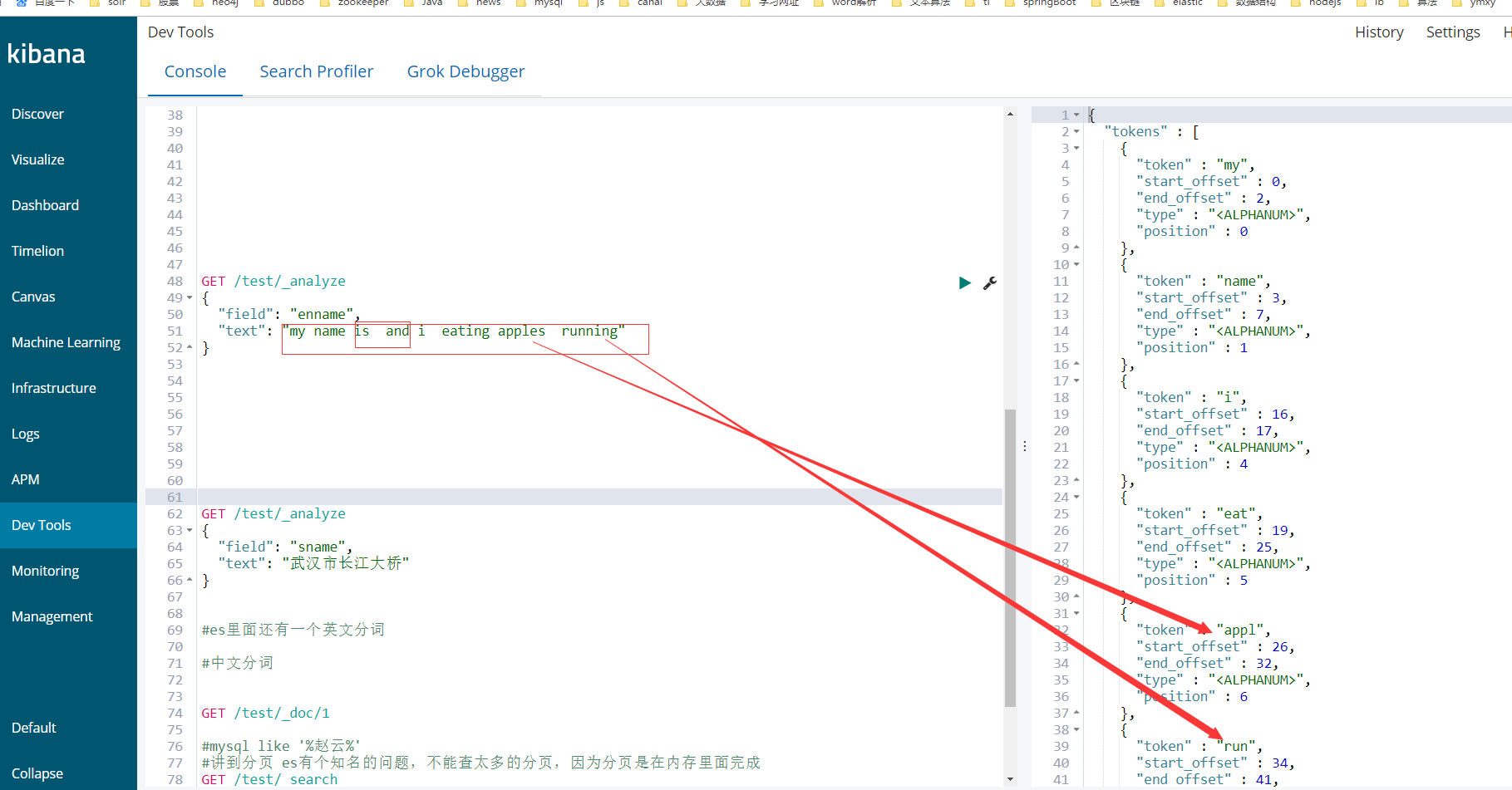
"query":{

"match": {"name":"eat"}

}

}

没搜到结果，我们使用后使用analyze api查看分析处理结果，可以看到没有分出eat，所以搜不到，改成用english分词器做。就可以看到以下的情况



可以看到running和apples被提取成了词干，提取算法其实就是一个映射感兴趣的同学可以自己取看看。而and 和 is被当作停用词去掉了。

（3）IK分词器：他会有两种模式一种smart，一种max\_word. 建索引的时候 用ik\_max\_word，搜索的时候用smart。如果需要详细理解里面的原理的，请看算法课的第23节课。讲的相当详细。

**分词的妙用：**

是不是就意味着stander分词没用了？其实并不是这样的哦

#托底，搜江大桥没有，在建了ik的字段，在建一个一样的stander的字段。如果ik搜不到 就可以搜这一个stander分词的，这样保证会又结果。但是慎用，因为占空间，有些特殊的系统可以使用。

#其实ik也还有一个解决的办法 叫砍词：江大桥 我可以砍掉一个词，我砍掉江 就出来了。砍词的策略可以自定义

#江大桥：电商中。我们系统假设有大桥这个品牌。 我会一个个的是去试一下，比如可以用字符串匹配找出大桥。也有很多系统很粗暴，直接从第一个字开始砍，一直砍到有为止。

#既有英文又有中文的 直接选ik

#如果不用砍词那就要去词库加词，比如加入江大桥就可以了。具体路径是在es的：

/apps/svr/elasticsearch-node1/config/analysis-ik/main.dic

注意集群的话那就要所有的es都需要加的哦