## 启动elasticsearch服务

到安装目录的bin下，service.bat start

(如果是刚安装完，先运行service.bat install)

## 查看服务状态

curl 'http://localhost:9200/?pretty'

pretty显示格式化后的json数据

## 访问Marvel管理

<http://localhost:9200/_plugin/marvel>

## 访问Sense开发者控制台

<http://localhost:9200/_plugin/marvel/sense/>

## 与elasticsearch交互

1. 节点客户端(node client)：

节点客户端以无数据节点(none data node)身份加入集群，换言之，它自己没有数据，但是知道什么数据位于集群的哪个节点上，能够直接转发请求到对应的节点上。

1. 传输客户端(Transport client)：

更轻量的传输客户端能够发送请求到远程集群，它自己不加入集群，只是简单转发请求给集群中的节点。

两个Java客户端都通过9300端口与集群交互，使用Elasticsearch传输协议(Elasticsearch Transport Protocol)。集群中的节点也通过9300端口通信。如果此端口未开放，你的节点将不能形成集群。

## elasticsearch与关系数据库对比



Elasticsearch集群可以包含多个索引(indices)（数据库），这些库可以包含多个类型(types)（表），这些类型包含多个文档(documents)（行），然后每个文档包含多个字段(Fields)（列）。

## 检索文档

GET /megacorp/employee/1

我们通过把HTTP方法PUT改为GET来检索文档，我们可以使用DELETE方法删除文档，使用HEAD方法检查文档是否存在。如果为了替换已存在的文档，我们只需使用PUT方法即可。

## 简单检索

GET /megacorp/employee/\_search

## 查询字符串搜索

GET /megacorp/employee/\_search?q=last\_name:Smith

last\_name是一个字段名字

## DSL语句

GET /megacorp/employee/\_search

{

"query": {

"match":{

"last\_name":"Smith"

}

}

}

## get和post的区别

get不能携带出url之外的数据，post可以额外提交数据

## 使用过滤器



## 全文检索



\_score Elasticsearch根据相关评分排序，相关评分是根据文档与语句的匹配度来得出

## 短语搜索



## 高亮显示搜索结果



## 聚合

允许在数据基础上生成复杂的统计



从匹配的结果中统计



分级汇总



## Java api

### 与集群交互

可以通过两种方式来连接到elasticsearch（简称es）集群，第一种是通过在你的程序中创建一个嵌入es节点（Node），使之成为es集群的一部分，然后通过这个节点来与es集群通信。第二种方式是用TransportClient这个接口和es集群通信。

#### Node方式

创建嵌入节点的方式如下：

import static org.elasticsearch.node.NodeBuilder.\*;

//启动节点

Node node = nodeBuilder().node();

Client client = node.client();

//关闭节点

node.close();

当你启动一个节点，它会自动加入同网段的es集群，一个前提就是es的集群名（cluster.name）这个参数要设置一致。

默认的话启动一个节点，es集群会自动给它分配一些索引的分片，如果你想这个节点仅仅作为一个客户端而不去保存数据，你就可以设置把node.data设置成false或 node.client设置成true。下面是例子：

Node node = nodeBuilder().clusterName(clusterName).client(true).node();

还有一种情况是你并不想把节点加入集群，只想用它进行单元测试时，就要启动一个“本地”的es，这里“本地”指的是在jvm的级别下运行，即两个不同的es节点运行在同一个JVM中时会组成一个集群。它需要把节点的local参数设置成true，下面是例子：

Node node = nodeBuilder().local(true).node();

#### TransportClient方式

通过TransportClient这个接口，我们可以不启动节点就可以和es集群进行通信，它需要指定es集群中其中一台或多台机的ip地址和端口，例子如下：

Client client = new TransportClient()

.addTransportAddress(new InetSocketTransportAddress("host1", 9300))

.addTransportAddress(new InetSocketTransportAddress("host2", 9300));

client.close();

如果你需要更改集群名（默认是elasticsearch），需要如下设置：

Settings settings = ImmutableSettings.settingsBuilder()

.put("cluster.name", "myClusterName").build();

Client client = new TransportClient(settings);

你可以设置client.transport.sniff为true来使客户端去嗅探整个集群的状态，把集群中其它机器的ip地址加到客户端中，这样做的好处是一般你不用手动设置集群里所有集群的ip到连接客户端，它会自动帮你添加，并且自动发现新加入集群的机器。代码实例如下：

Settings settings = ImmutableSettings.settingsBuilder()

.put("client.transport.sniff", true).build();

TransportClient client = new TransportClient(settings);

### put Mapping定义索引字段属性

Mapping,就是对索引库中索引的字段名及其数据类型进行定义，类似于关系数据库中表建立时要定义字段名及其数据类型那样，不过es的mapping比数据库灵活很多，它可以动态添加字段。一般不需要要指定mapping都可以，因为es会自动根据数据格式定义它的类型，如果你需要对某些字段添加特殊属性（如：定义使用其它分词器、是否分词、是否存储等），就必须手动添加mapping。有两种添加mapping的方法，一种是定义在配置文件中，一种是运行时手动提交mapping，两种选一种就行了。

先介绍在配置文件中定义mapping，你可以把[mapping名].json文件放到config/mappings/[索引名]目录下，这个目录要自己创建，一个mapping和一个索引对应，你也可以定义一个默认的mapping，把自己定义的default-mapping.json放到config目录下就行。json格式如下：

{

"mappings":{

"properties":{

"title":{

"type":"string",

"store":"yes"

},

"description":{

"type":"string",

"index":"not\_analyzed"

},

"price":{

"type":"double"

},

"onSale":{

"type":"boolean"

},

"type":{

"type":"integer"

},

"createDate":{

"type":"date"

}

}

}

}

接下来介绍通过请求添加mapping，下面为一个添加productIndex索引库的mapping的json格式请求。其中productIndex为索引类型，properties下面的为索引里面的字段，type为数据类型，store为是否存储，"index":"not\_analyzed"为不对该字段进行分词。

{

"productIndex":{

"properties":{

"title":{

"type":"string",

"store":"yes"

},

"description":{

"type":"string",

"index":"not\_analyzed"

},

"price":{

"type":"double"

},

"onSale":{

"type":"boolean"

},

"type":{

"type":"integer"

},

"createDate":{

"type":"date"

}

}

}

}

用java api调用的代码如下：

先创建空索引库

client.admin().indices().prepareCreate("productIndex").execute().actionGet();

put mapping：

XContentBuilder mapping = jsonBuilder()

.startObject()

.startObject("productIndex")

.startObject("properties")

.startObject("title").field("type", "string").field("store", "yes").endObject()

.startObject("description").field("type", "string").field("index", "not\_analyzed").endObject()

.startObject("price").field("type", "double").endObject()

.startObject("onSale").field("type", "boolean").endObject()

.startObject("type").field("type", "integer").endObject()

.startObject("createDate").field("type", "date").endObject()

.endObject()

.endObject()

.endObject();

PutMappingRequest mappingRequest = Requests.putMappingRequest("productIndex").type("productIndex").source(mapping);

client.admin().indices().putMapping(mappingRequest).actionGet();

### 索引数据

es索引数据非常方便，只需构建个json格式的数据提交到es就行，下面是个java api的例子

XContentBuilder doc = jsonBuilder()

.startObject()

.field("title", "this is a title!")

.field("description", "descript what?")

.field("price", 100)

.field("onSale", true)

.field("type", 1)

.field("createDate", new Date())

.endObject();

client.prepareIndex("productIndex","productType").setSource(doc).execute().actionGet();

其中productIndex为索引库名，一个es集群中可以有多个索引库。productType为索引类型，是用来区分同索引库下不同类型的数据的，一个索引库下可以有多个索引类型。

### 删除索引数据

删除api允许从特定索引通过id删除json文档。有两种方法，一是通过id删除，二是通过一个Query查询条件删除，符合这些条件的数据都会被删除。

一、通过id删除

下面的例子是删除索引名为twitter，类型为tweet，id为1的文档：

DeleteResponse response = client.prepareDelete("twitter", "tweet", "1")

.execute()

.actionGet();

二、通过Query删除

下面的例子是删除索引名为productIndex，title中包含query的所有文档：

QueryBuilder query = QueryBuilders.fieldQuery("title", "query");

client.prepareDeleteByQuery("productIndex").setQuery(query).execute().actionGet();

设置线程

当删除api在同一个节点上执行时（在一个分片中执行一个api会分配到同一个服务器上），删除api允许执行前设置线程模式（operationThreaded选项），operationThreaded这个选项是使这个操作在另外一个线程中执行，或在一个正在请求的线程（假设这个api仍是异步的）中执行。默认的话operationThreaded会设置成true，这意味着这个操作将在一个不同的线程中执行。下面是设置成false的方法：

DeleteResponse response = client.prepareDelete("twitter", "tweet", "1")

.setOperationThreaded(false)

.execute()

.actionGet();

官方文档：

http://www.elasticsearch.org/guide/reference/api/delete.html

http://www.elasticsearch.org/guide/reference/java-api/delete.html

### 搜索

elasticsearch的查询是通过执行json格式的查询条件，在java api中就是构造QueryBuilder对象，elasticsearch完全支持queryDSL风格的查询方式，QueryBuilder的构建类是QueryBuilders，filter的构建类是FilterBuilders。下面是构造QueryBuilder的例子：

import static org.elasticsearch.index.query.FilterBuilders.\*;

import static org.elasticsearch.index.query.QueryBuilders.\*;

QueryBuilder qb1 = termQuery("name", "kimchy");

QueryBuilder qb2 = boolQuery()

.must(termQuery("content", "test1"))

.must(termQuery("content", "test4"))

.mustNot(termQuery("content", "test2"))

.should(termQuery("content", "test3"));

QueryBuilder qb3 = filteredQuery(

termQuery("name.first", "shay"),

rangeFilter("age")

.from(23)

.to(54)

.includeLower(true)

.includeUpper(false)

);

其中qb1构造了一个TermQuery，对name这个字段进行项搜索，项是最小的索引片段，这个查询对应lucene本身的TermQuery。 qb2构造了一个组合查询（BoolQuery），其对应lucene本身的BooleanQuery，可以通过must、should、mustNot方法对QueryBuilder进行组合，形成多条件查询。 qb3构造了一个过滤查询，就是在TermQuery的基础上添加一个过滤条件RangeFilter，这个范围过滤器将限制查询age字段大于等于23，小于等于54的结果。除了这三个，elasticsearch还支持很多种类的查询方式，迟点写个介绍。

构造好了Query就要传到elasticsearch里面进行查询，下面是例子：

SearchResponse response = client.prepareSearch("test")

.setQuery(query)

.setFrom(0).setSize(60).setExplain(true)

.execute()

.actionGet();

这句的意思是，查询test索引，查询条件为query，从第0条记录开始，最多返回60条记录。返回结果为SearchResponse，下面解析SearchResponse：

SearchHits hits = searchResponse.hits();

for (int i = 0; i < 60; i++) {

System.out.println(hits.getAt(i).getSource().get("field"));

}

获得SearchResponse中的SearchHits，然后hits.getAt(i).getSource().get("field")获得field字段的值。

### 批量添加索引

elasticsearch支持批量添加或删除索引文档，java api里面就是通过构造BulkRequestBuilder，然后把批量的index/delete请求添加到BulkRequestBuilder里面，执行BulkRequestBuilder。下面是个例子：

import static org.elasticsearch.common.xcontent.XContentFactory.\*;

BulkRequestBuilder bulkRequest = client.prepareBulk();

bulkRequest.add(client.prepareIndex("twitter", "tweet", "1")

.setSource(jsonBuilder()

.startObject()

.field("user", "kimchy")

.field("postDate", new Date())

.field("message", "trying out Elastic Search")

.endObject()

)

);

bulkRequest.add(client.prepareIndex("twitter", "tweet", "2")

.setSource(jsonBuilder()

.startObject()

.field("user", "kimchy")

.field("postDate", new Date())

.field("message", "another post")

.endObject()

)

);

BulkResponse bulkResponse = bulkRequest.execute().actionGet();

if (bulkResponse.hasFailures()) {

//处理错误

}