Elasticsearch 分布式搜索 https://www.elastic.co/cn/

1. elasticsearch是一款非常强大的开源搜索引擎，可以帮助我们从海量数据中快速找到需要的内容。

京东根据关键字搜索有关商品,百度搜索和关键字有关的信息,滴滴打车搜索出离你最近的车等等 都可以用elasticsearch实现

一个开源的分布式搜索引擎，可以用来实现搜索、日志统计、分析、系统监控等功能

1. elasticsearch结合kibana、Logstash、Beats，也就是elastic stack（ELK）。被广泛应用在微服务的日志数据分析、实时监控等领域。

是以elasticsearch为核心的技术栈，包括beats、Logstash、kibana、elasticsearch

实时监控:

项目自运行的过程中会产生海量的日志信息,定位我们系统问题的信息;运行的状态也是数据,比如cpu,内存信息,访问量等等;都会被elasticsearch所管理起来,也就是实时监控

1. 几大组件:

核心:elasticsearch负责数据的存储,计算,分析,搜索信息等;

Logstash和Beats负责数据抓取,比如日志信息;

Kibana是一个数据可视化的组件,比如展示的话用它展示成报表

Kibana和Logstash和Beats都是可替代的,不可替代的elasticsearch核心 底层实现Lucene

Lucene是一个Java语言的搜索引擎类库，是Apache公司的顶级项目，由DougCutting于1999年研发 官网:https://lucene.apache.org/

类库也就是jar包

Lucene的优势：

易扩展 可以基于它做二次的开发和定制,实现更高级的功能

高性能（基于倒排索引）查询性能非常好

Lucene的缺点：

只限于Java语言开发 因为是类库(java)

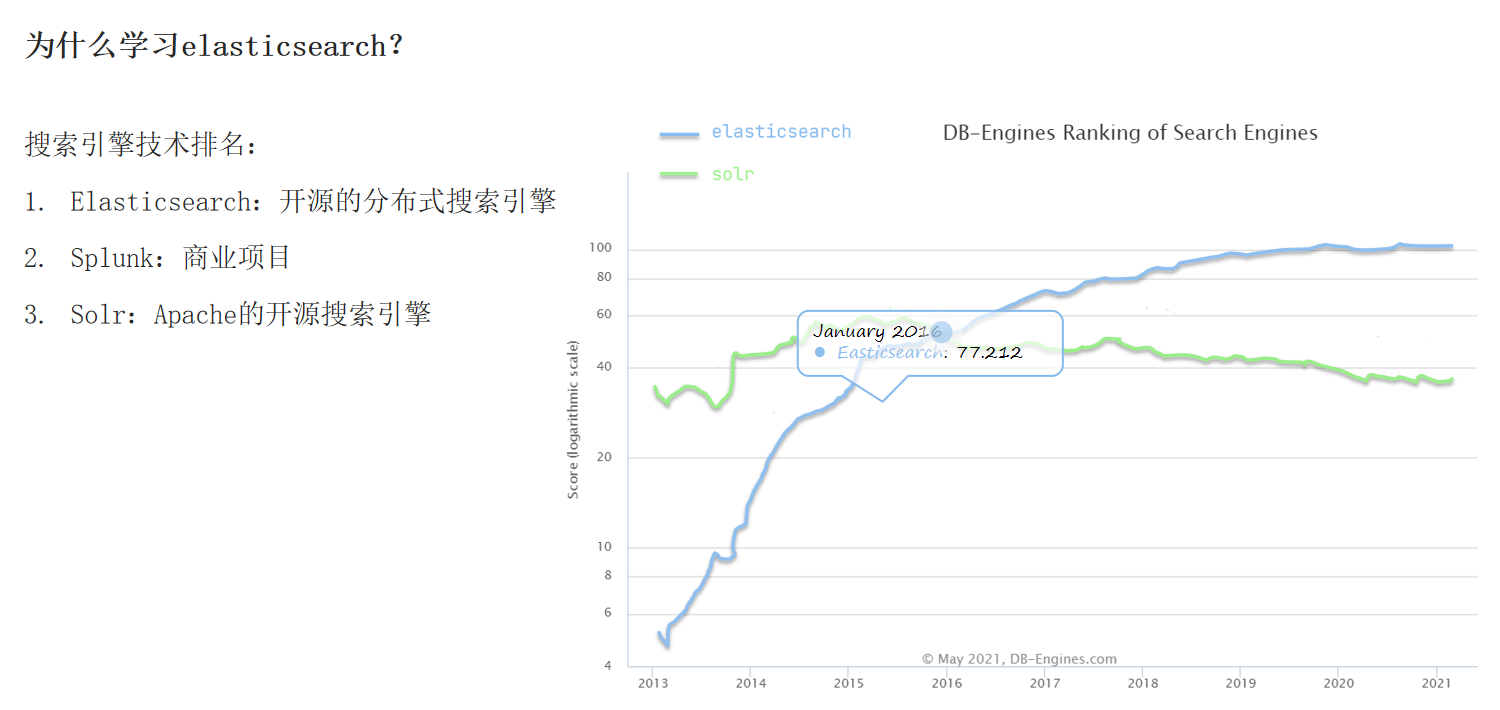
学习曲线陡峭

不支持水平扩展

elasticsearch基于Lucene做的二次开发 2010Shay Banon开发的 天生支持集群和分布式

支持分布式，可水平扩展 脱离了语言的限制

提供Restful接口，可被任何语言调用



Lucene中的核心技术倒排索引

基于传统数据库比如mysql对比得出的名字 传统数据库（如MySQL）采用正向索引，例如给下表（tb\_goods）中的id创建索引

一般都是给id创建索引形成一颗b+树,依据id检索会非常的快,这种就叫正向索引,如果搜索的是标题,有没有包涵手机的那就需要用前置模糊查询了,会不走索引,就会一条一条扫有没有包涵手机,如果没有丢弃,如果有放入到标题,正向索引查局部内容的时候一条一条扫性能会很差



elasticsearch采用倒排索引：

文档（document）：每条数据就是一个文档

词条（term）：文档按照语义分成的词语 比如分成:手机,华为,小米,手环等 每个词就是一个磁条

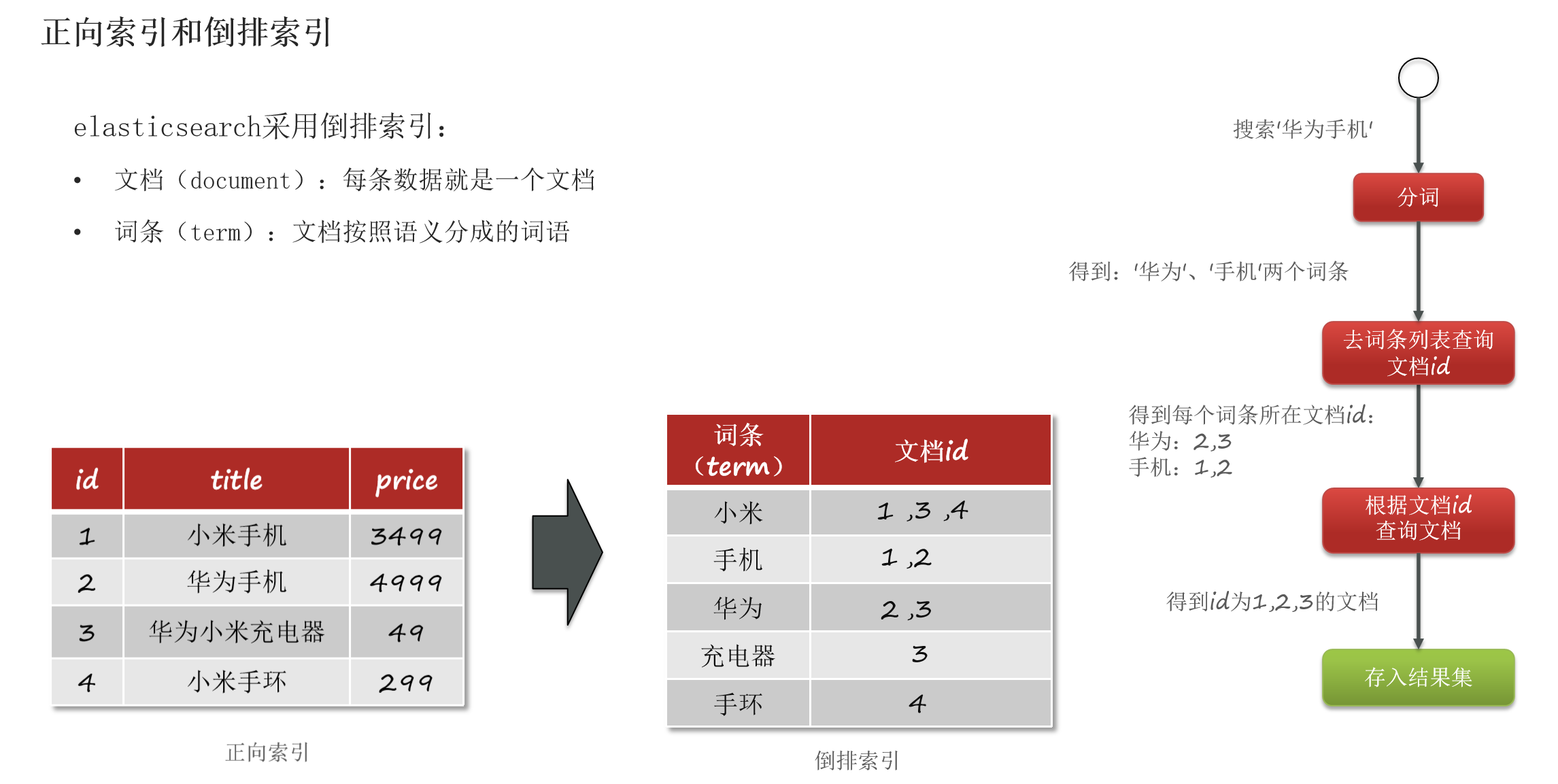
倒排索引创建的时候会创建成一张新的表一个字段是磁条一个字段是文档id

倒排索引存储的时候会先按照磁条去存,比如要对标题进行分词小米和手机分别是一个新的磁条,然后再把id保存下来

不管是多少条数据 都是分成一个个磁条,每个词条会有大量的重复,记录磁条和对应的文档的id,确保磁条不重复,我们就可以对磁条创建索引,可以使用hash法或者是b+树创建唯一索引,将来依据此条查询的速度会非常的快

搜索时会对用户输入的进行分词,得到不同磁条,再用词条去查询,因为词条上有id所以查询很快,得到几组文档id,再用id快速定位得到文档所有内容

普通索引依据文档找到词,倒排索引是先找到词在找到文档,基于内容去搜索





什么是文档和词条？

每一条数据就是一个文档

词条对文档中的内容分词，得到的词语就是词条

什么是正向索引？

基于文档id创建索引。查询词条时必须先找到文档，而后判断是否包含词条 非索引字段逐行扫描

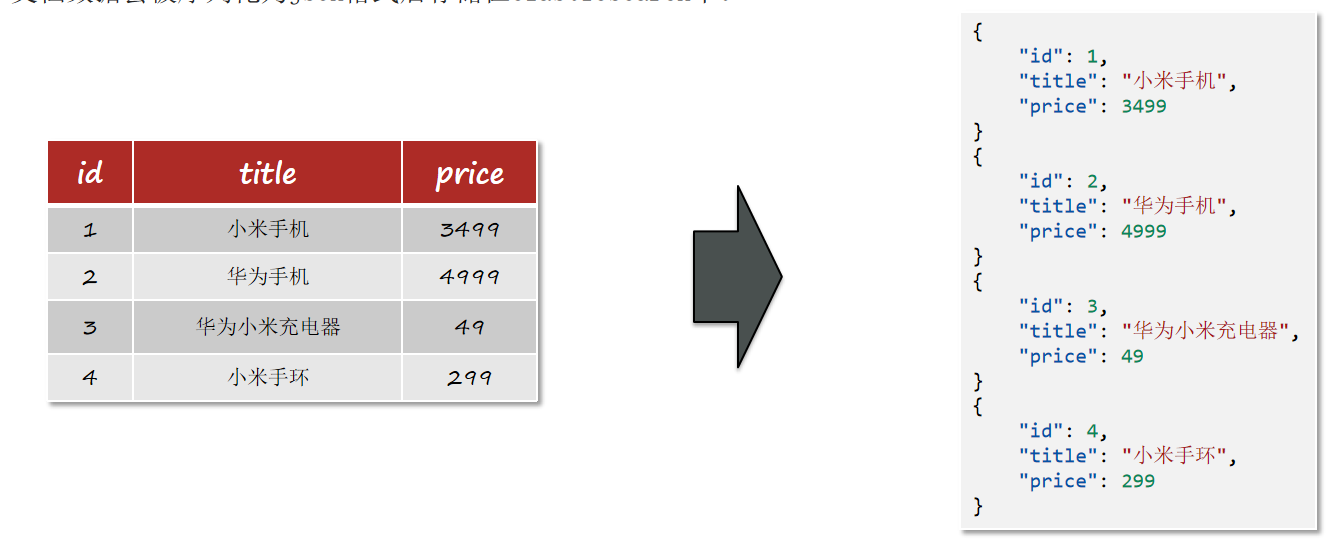
什么是倒排索引？

对文档内容分词，对词条创建索引，并记录词条所在文档的信息。查询时先根据词条查询到文档id，而后获取到文档

1. Es和数据库对比

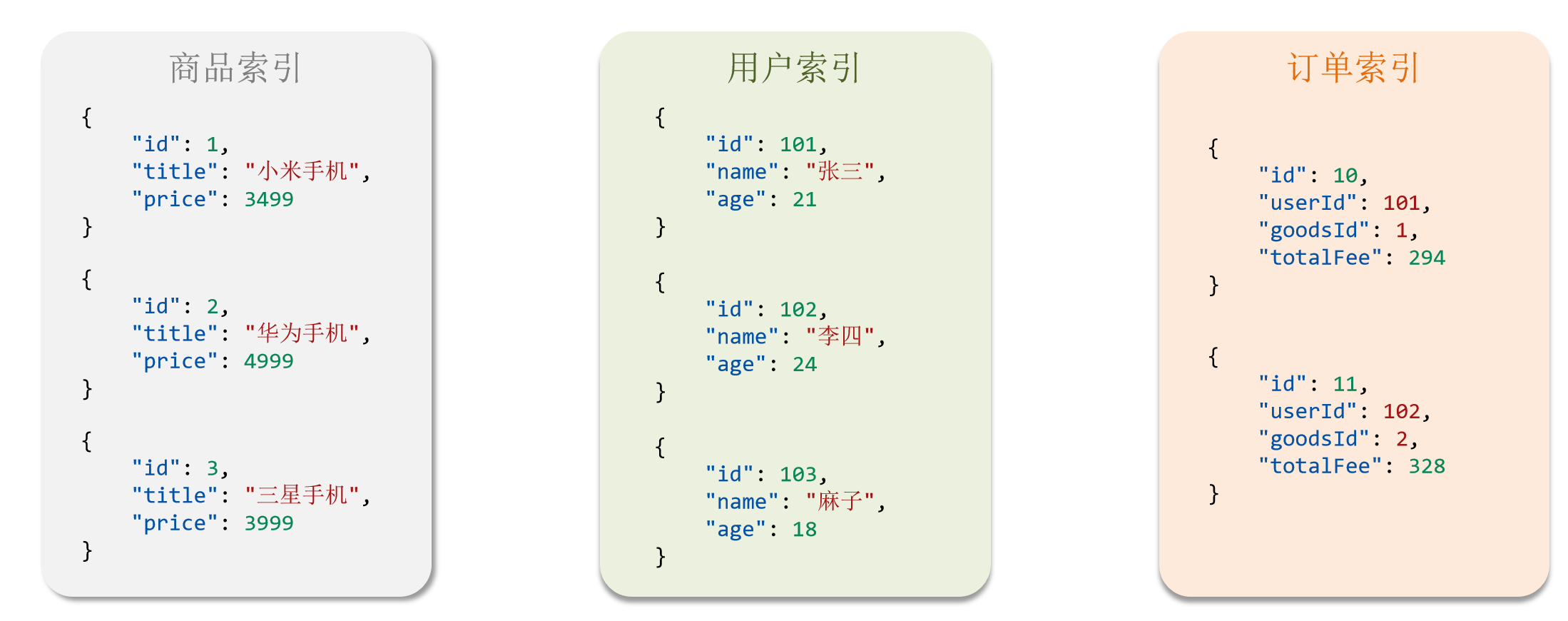
(1)elasticsearch是面向文档存储的，可以是数据库中的一条商品数据，一个订单信息。

订单也好标题也好都会变成文档,文档数据会被序列化为json格式后存储在elasticsearch中。



1. 索引（index）：相同类型的文档的集合,划分不同类型,放到不同的索引库 同种类型的文档叫做索引

映射（mapping）：索引中文档的字段约束信息，类似表的结构约束 类型约束



概念对比

Table表对应index索引 table每一条数据的集合 index每一个文档的集合

Row每行数据对应文档document

Column列对应filed字段 转完json变成一个一个字段

Schema约束(字段名称,数据类型)对应Mapping映射 每个索引库都会有自己的映射就像创建表一样

Sql语句对应dsl语句 语法 用来对文档的操作 json风格的请求语句的 基于http请求发送出去的 rest福利风格 好处:与语言无关,脱离语言的束缚



架构 mysql和Elasticsearch互补而不是替代,两个都会存在

elasticsearch与数据库的关系：

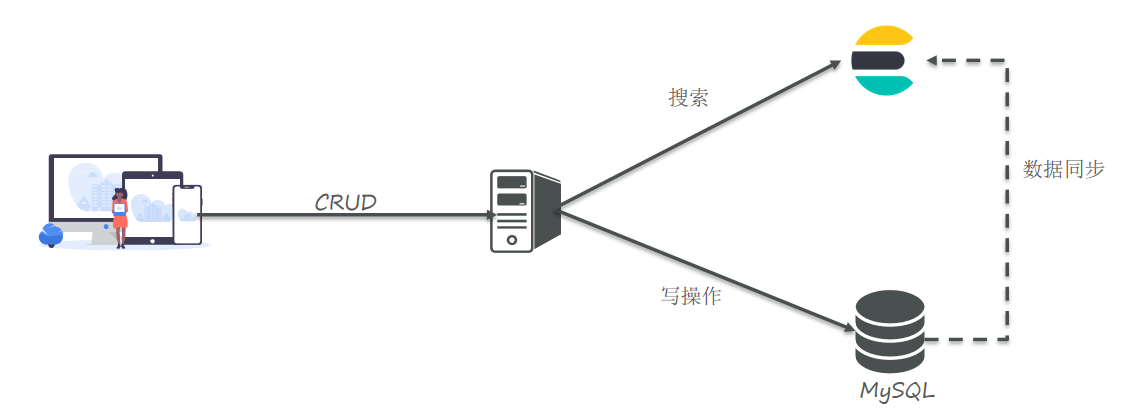
数据库负责事务类型操作

elasticsearch负责海量数据的搜索、分析、计算

Mysql：擅长事务类型操作，可以确保数据的安全和一致性 (如果是下单业务事务要求高,用mysql)

Elasticsearch：擅长海量数据的搜索、分析、计算 (如果是页面搜索要求高,用Elasticsearch)

用户发送个请求到服务器判断是增删改操作,写入到mysql;如果是查询去es做搜索;写操作的时候mysql将数据同步到es,实现双写



安装es和kibana

Kibana提供了demotwos的一个工具,方便编写dsl语句从而去操作es

# 部署单点es

## **创建网络**

因为我们还需要部署kibana容器，因此需要让es和kibana容器互联。这里先创建一个网络：

docker network create es-net

加载es镜像 上传到虚拟机 load导入

docker run -d \ #后台运行

--name es \ #容器名

-e "ES\_JAVA\_OPTS=-Xms512m -Xmx512m" \ #设置环境变量 jvm堆内存大小 默认1g太大

-e "discovery.type=single-node" \ #设置环境变量 单点模式 做集群时候要改

-v es-data:/usr/share/elasticsearch/data \ #数据卷挂载目录 数据保存目录

-v es-plugins:/usr/share/elasticsearch/plugins \ #数据卷挂载目录 插件目录

--privileged \

--network es-net \ #加入到网络当中

-p 9200:9200 \ #暴露端口 供http用户访问用的

-p 9300:9300 \ #暴露端口 各个节点连接的端口

elasticsearch:7.12.1 #镜像名和版本

此时访问宿主机ip和9200端口 访问出json格式的东西 容器运行成功

加载kibana镜像 上传到虚拟机 load导入

创建容器

docker run -d \

--name kibana \ #容器名

-e ELASTICSEARCH\_HOSTS=http://es:9200 \ #指定es地址 这个es就是es容器名的名称,因为在同一网络,所以可以使用容器名互联

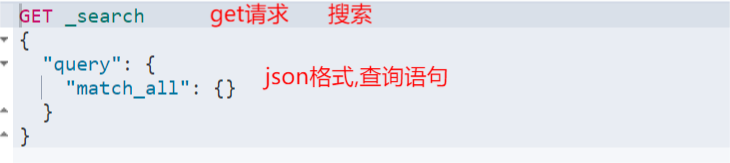
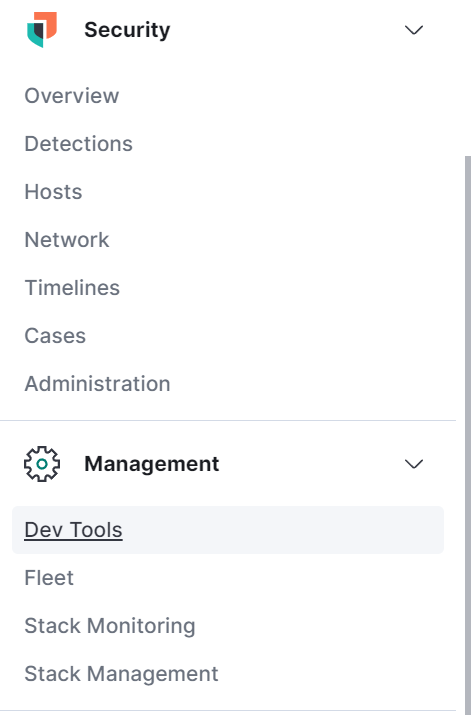
--network=es-net \ #网络

-p 5601:5601 \ #暴露端口

kibana:7.12.1 #镜像名和版本 一定要与es版本一样

此时访问宿主机5601端口 就可以看到kibana界面了 进入dsl语句操作界面

本质作用 发送一个dsl语句到es中 查询



模拟请求restful风格 因为kibana用es端口去访问 如果访问[192.168.56.128:9200](http://192.168.56.128:9200/)就是为空 ,那么我们直接在kibana

发送一个get /发现和访问[192.168.56.128:9200](http://192.168.56.128:9200/)得到的数据是一样的



1. Es中分词器

es在创建倒排索引时需要对文档分词；在搜索时，需要对用户输入内容分词。但默认的分词规则(standard)对中文处理并不友好。

执行如下 发现对英语分词还好,但是对中文进行逐字分词

分词器类型:standard(默认) english Chinese 发现每个分词都是这样的 没有办法理解中文意思按照内容分



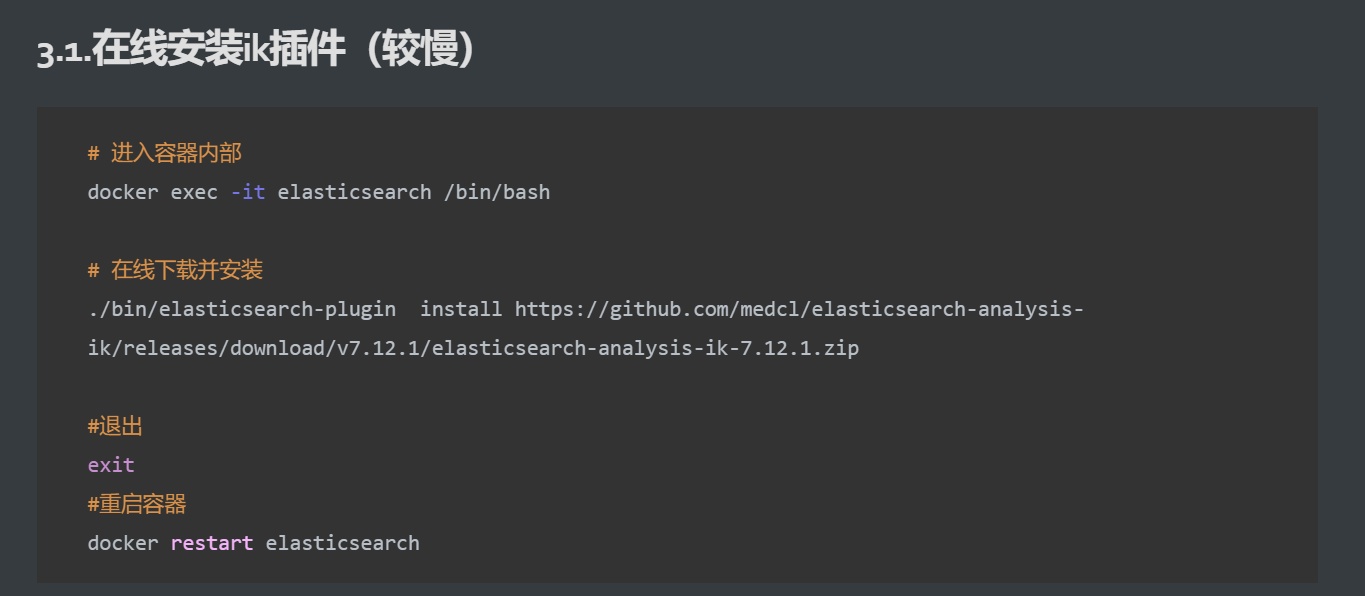
解决办法 处理中文分词，一般会使用IK分词器。https://github.com/medcl/elasticsearch-analysis-ik

有两种分词器 ik\_smart：最少切分，粗粒度

ik\_max\_word：最细切分，细粒度

安装:两种办法

第一种(慢不推荐):



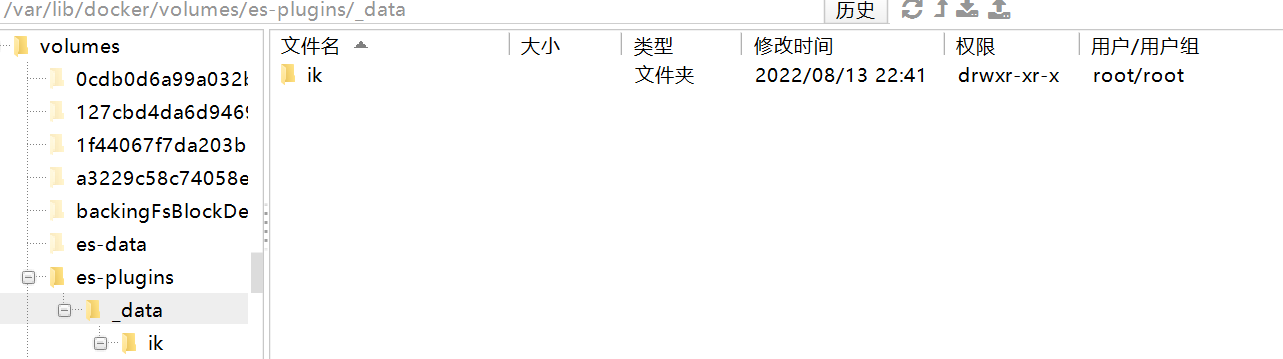
第二种

第一步:安装插件要知道es中plugins挂载的目录,而我们用了数据卷挂载,用以下方式查看



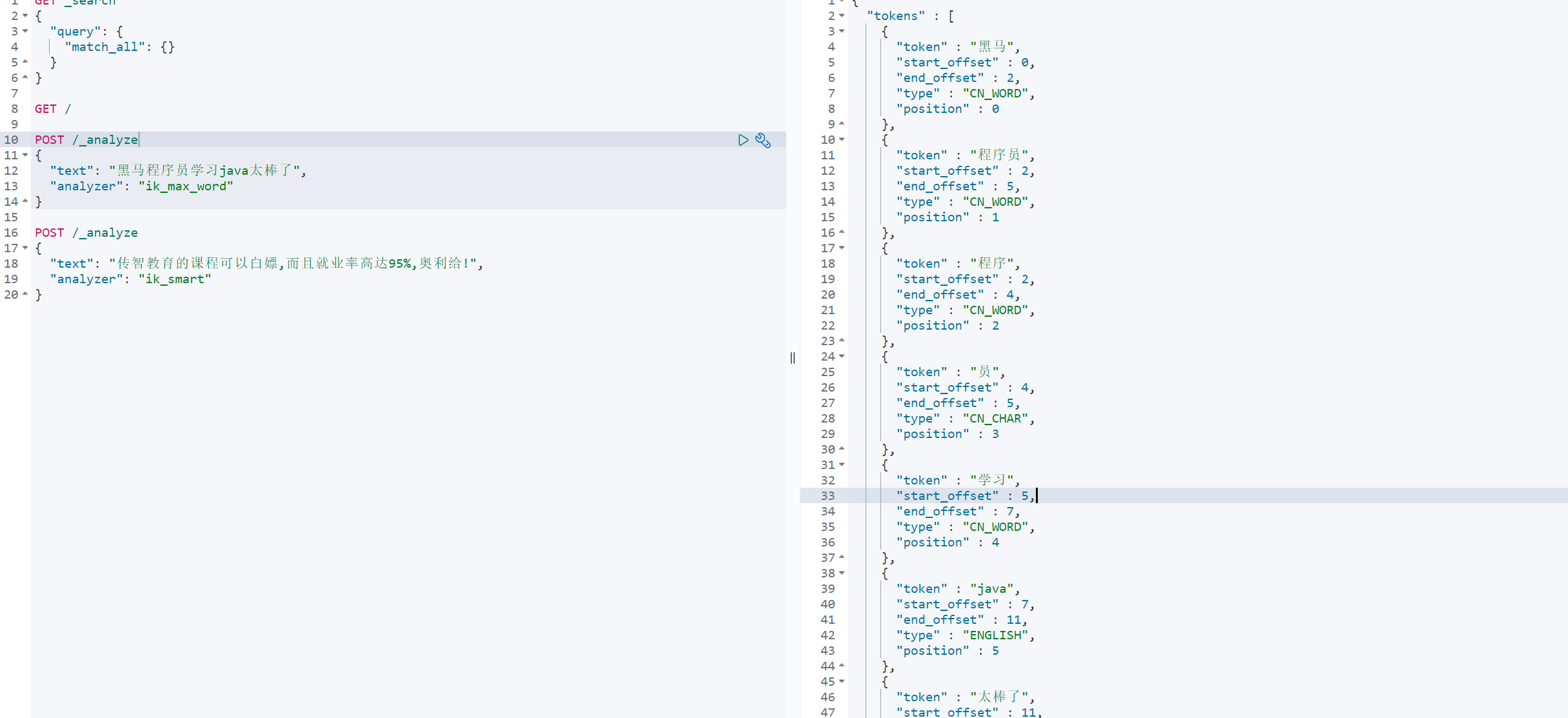
第二步:进入这个地址/var/lib/docker/volumes/es-plugins/\_data数据卷地址

将压缩包解压 将ik目录拖拽上传到这个目录



第三步:重启容器 docker restart es 查看日志 docker logs -f es

安装好进行测试 查看两种分词器区别 ik\_smart：最少切分，粗粒度 ik\_max\_word：最细切分，细粒度,内存消耗多

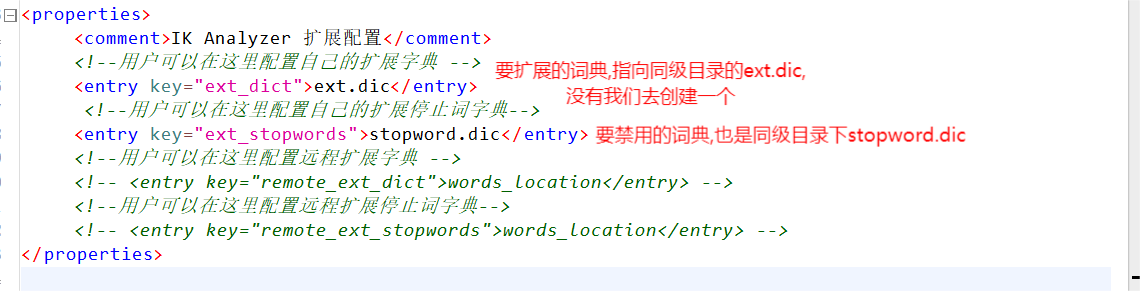


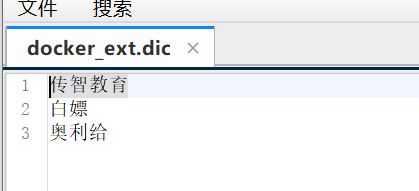


分词器底层是一个字典,对输入的文本进行分词,如果字典有给他分出来,但是像网络用语等(比如:奥利给,白嫖)词典中没有,所以我们要拓展ik分词器的词库,还有像敏感词和无用词我们要去停止(比如毒品,的,了,啊等)我们要去禁用

我们要对es配置文件进行修改/var/lib/docker/volumes/es-plugins/\_data/ik/config/IKAnalyzer.cfg.xml

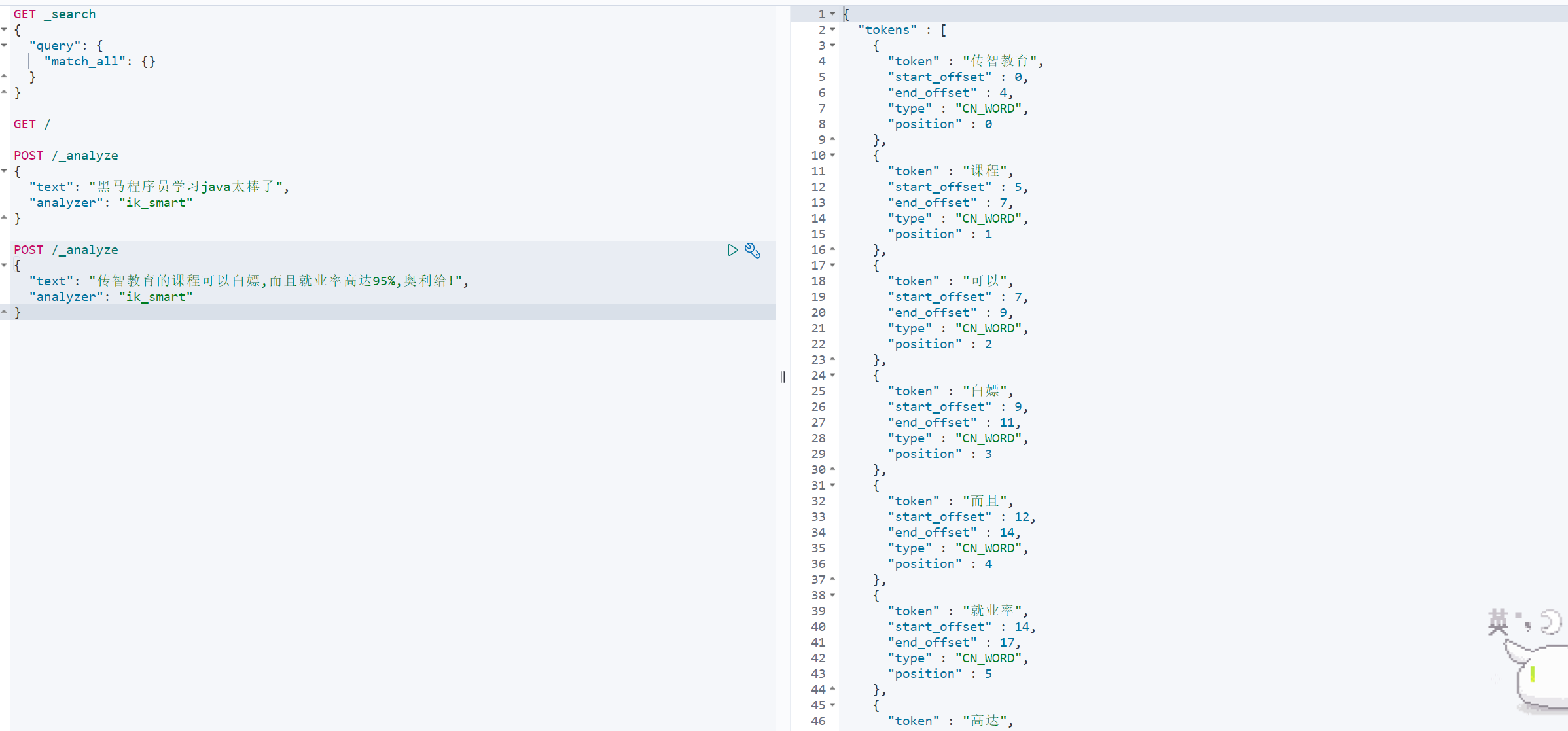
配置统计目录下如下两个文件 在这两个文件中我们可以去配置扩展的词和禁用的词





重启生效 docker restart es

再次进行分词发现我们的 传智教育和白嫖已经成了一个词 并且 的 无用词已经被排除掉了



分词器的作用是什么？

创建倒排索引时对文档分词 将文档创建倒排索引的时候对文档进行分词

用户搜索时，对输入的内容分词

IK分词器有几种模式？

ik\_smart：智能切分，粗粒度

ik\_max\_word：最细切分，细粒度

IK分词器如何拓展词条？如何停用词条？

利用config目录的IkAnalyzer.cfg.xml文件添加拓展词典和停用词典

在词典中添加拓展词条或者停用词条

1. 索引库的操作

索引库操作有哪些？

创建索引库：PUT /索引库名

查询索引库：GET /索引库名

删除索引库：DELETE /索引库名

添加字段：PUT /索引库名/\_mapping

创建一个索引库就像建表一样,建表的时候要设置每个字段的属性等 ,索引库也是一样的要设置mapping映射属性:对文档的约束

mapping是对索引库中文档的约束，常见的mapping属性包括：

type：字段数据类型，常见的简单类型有：

字符串：text（可分词的文本）、keyword（精确值，例如：品牌、国家、ip地址,不能拆的拆了没有意义!）

数值：long、integer、short、byte、double、float、 因为es底层是java实现的

布尔：boolean

日期：date

对象：object

没有数组这种类型,但是允许某一类型的字段有多个值比如数值类型有多个值

index：是否创建索引，默认为true 要不要创建倒排索引,没有倒排索引就没有办法去搜索这个字段,所以要想好这个字段要不要参加搜索

analyzer：使用哪种分词器 默认strant 使用ik\_smart或 ik\_max\_word

properties：该字段的子字段 如果像下面的object类型 用properties设定子属性

比如{

     "age": 21, #int

     "weight": 52.1, #double

     "isMarried": false, #boolean

     "info": "黑马程序员Java讲师", #text

"email": ["zy@itcast.cn",](mailto:\"zy@itcast.cn\",) #keyword

"score": [99.1, 99.5, 98.9], #double类型多个

    "name": { #object

         "firstName": "云", #子属性

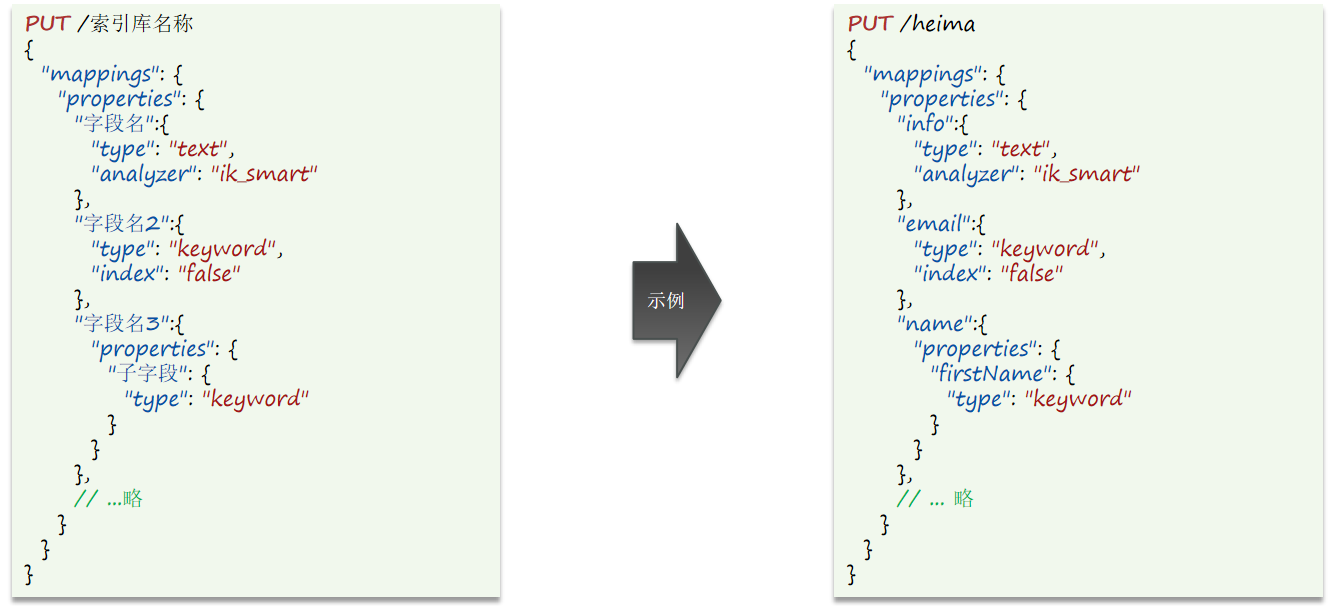
         "lastName": "赵" #子属性

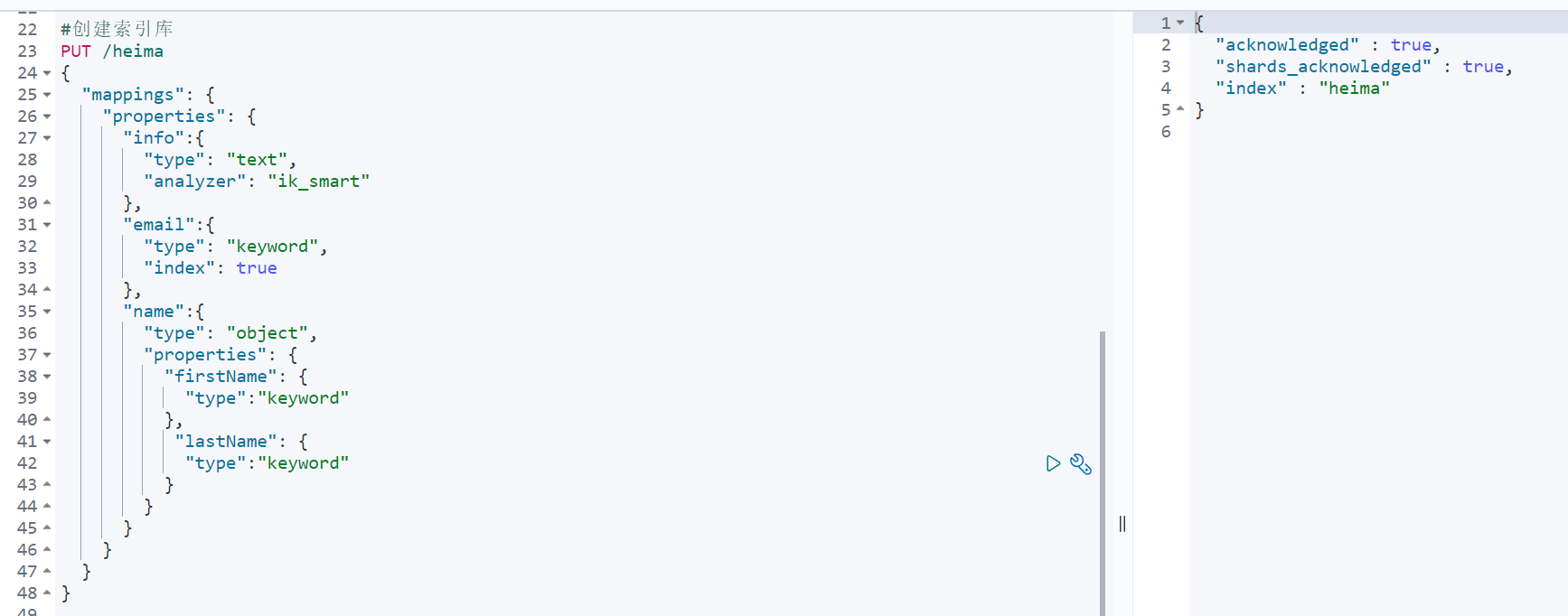
    }

}

创建索引库 dsl语句格式

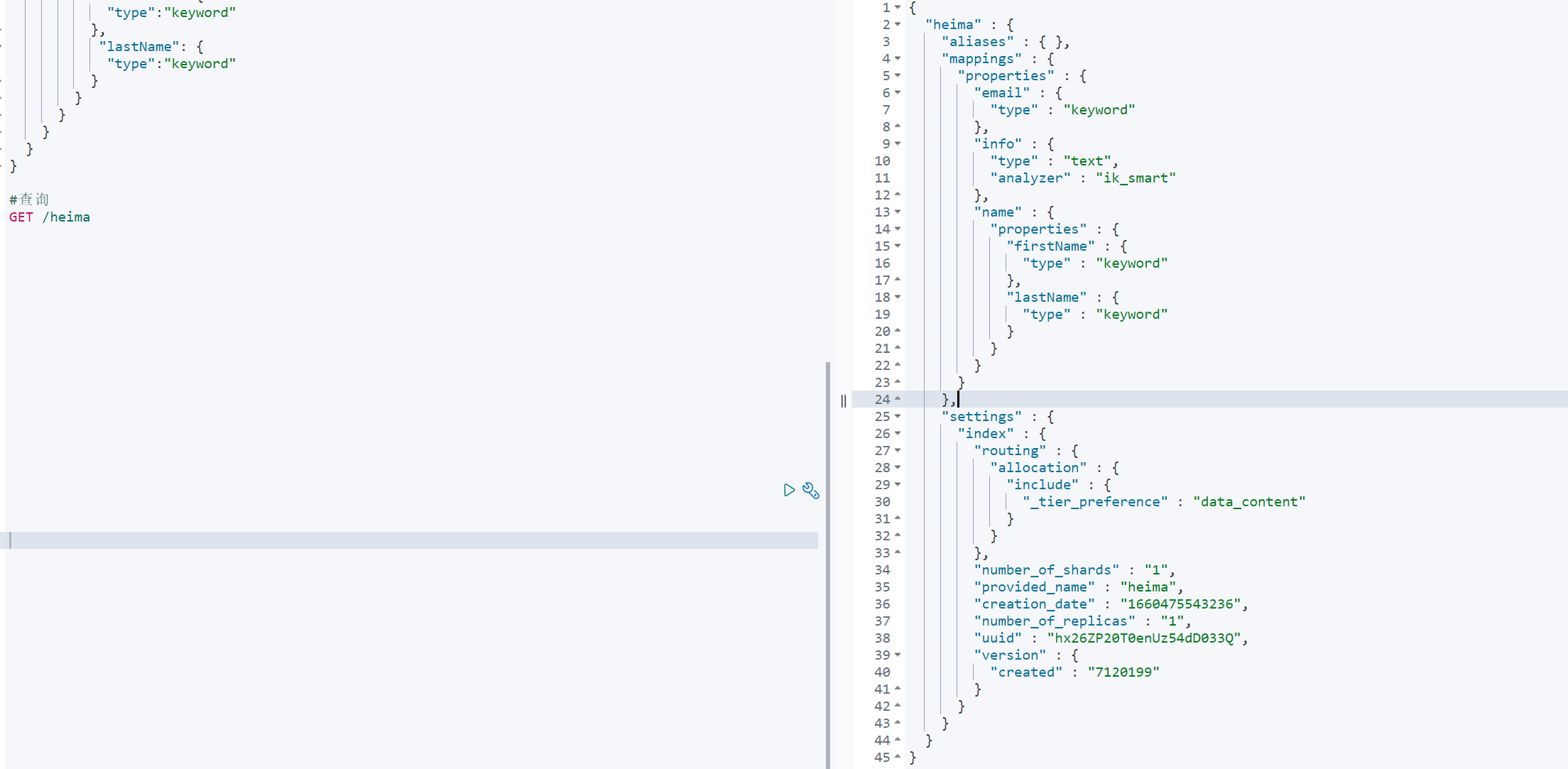
ES中通过Restful请求操作索引库、文档。请求内容用DSL语句来表示。创建索引库和mapping的DSL语法如下：





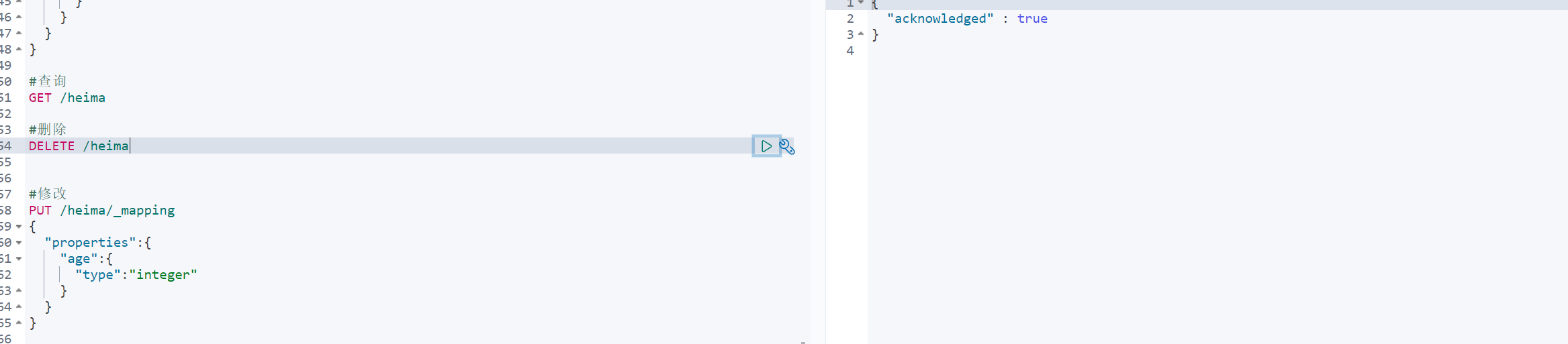
查询索引库

GET /索引库名



删除索引库的语法：

DELETE /索引库名



索引库是不允许修改的,索引库创建好之后数据结构已经创建好了,mapping映射基于这个创建倒排索引,如果修改的话,原先那些倒排索引就会失效,是禁止修改索引库的,但是允许添加新的字段

PUT /索引库名/\_mapping #代表我需要修改这个索引库的映射

{

  "properties": {

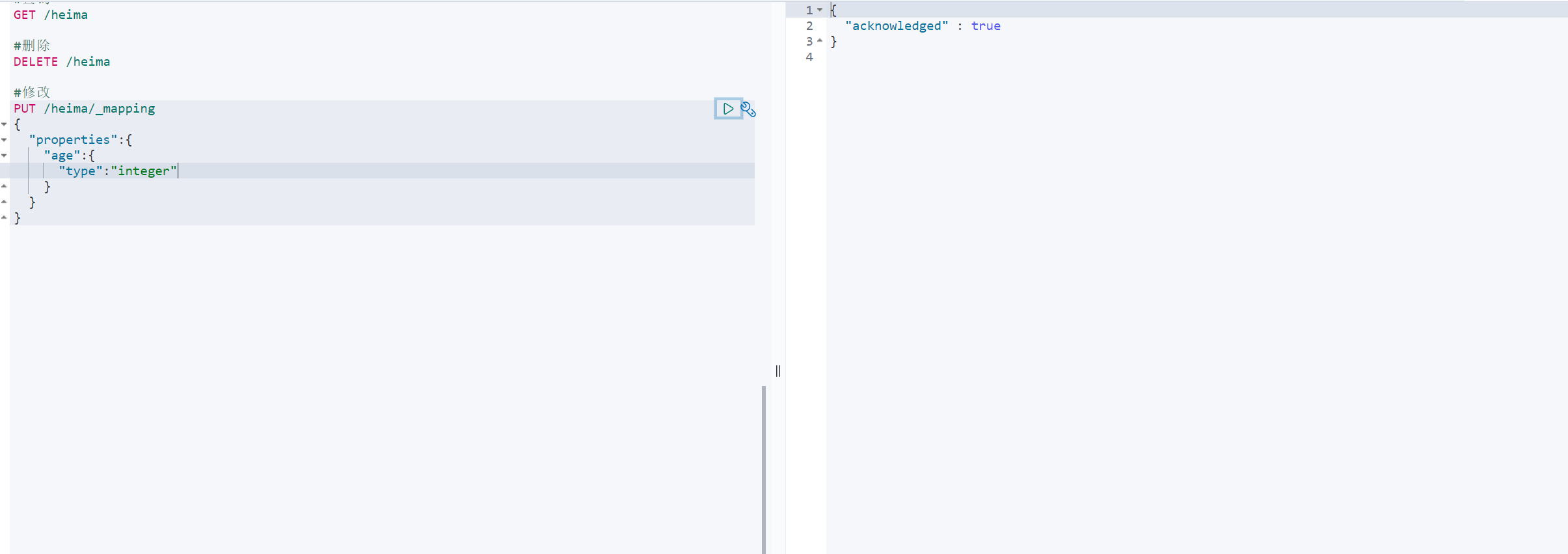
    "新字段名":{ #一定是一个全新的字段名 ,否则就是修改原有字段,就会报错

      "type": "integer" #类型

     }

  }

}



1. 文档操作

创建文档：POST /索引库名/\_doc/文档id { json文档 }

查询文档：GET /索引库名/\_doc/文档id

删除文档：DELETE /索引库名/\_doc/文档id

修改文档：

全量修改：PUT /索引库名/\_doc/文档id { json文档 }

增量修改：POST /索引库名/\_update/文档id { "doc": {字段}}

新增文档dsl语句 注意:文档id如果不加es会默认给个id ,这不是我们要的结果,一定要加

POST /索引库名/\_doc/文档id

{

    "字段1": "值1",

    "字段2": "值2",

    "字段3": {

        "子属性1": "值3",

        "子属性2": "值4"

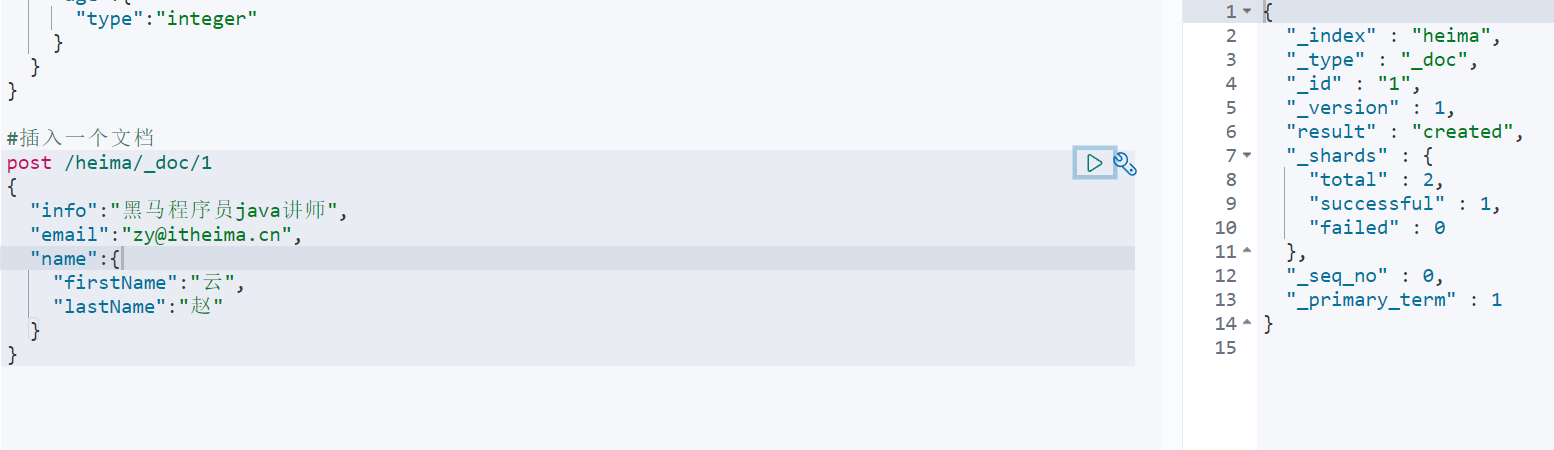
    },

// ...

}



与索引库字段名一致

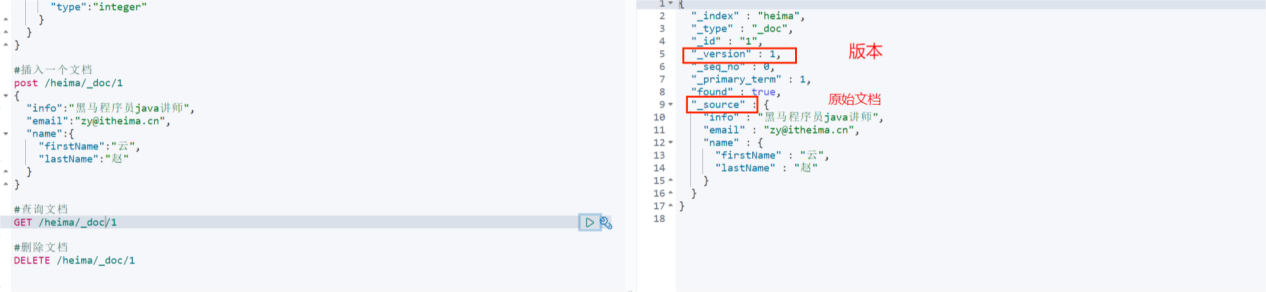


查看文档语法

GET /索引库名/\_doc/文档id

删除文档语法

Delete /索引库名/\_doc/文档id



注意version 每次变动都会加一 比如插入:1 删除变为2 再插入变为3

Source为插入的原始文档

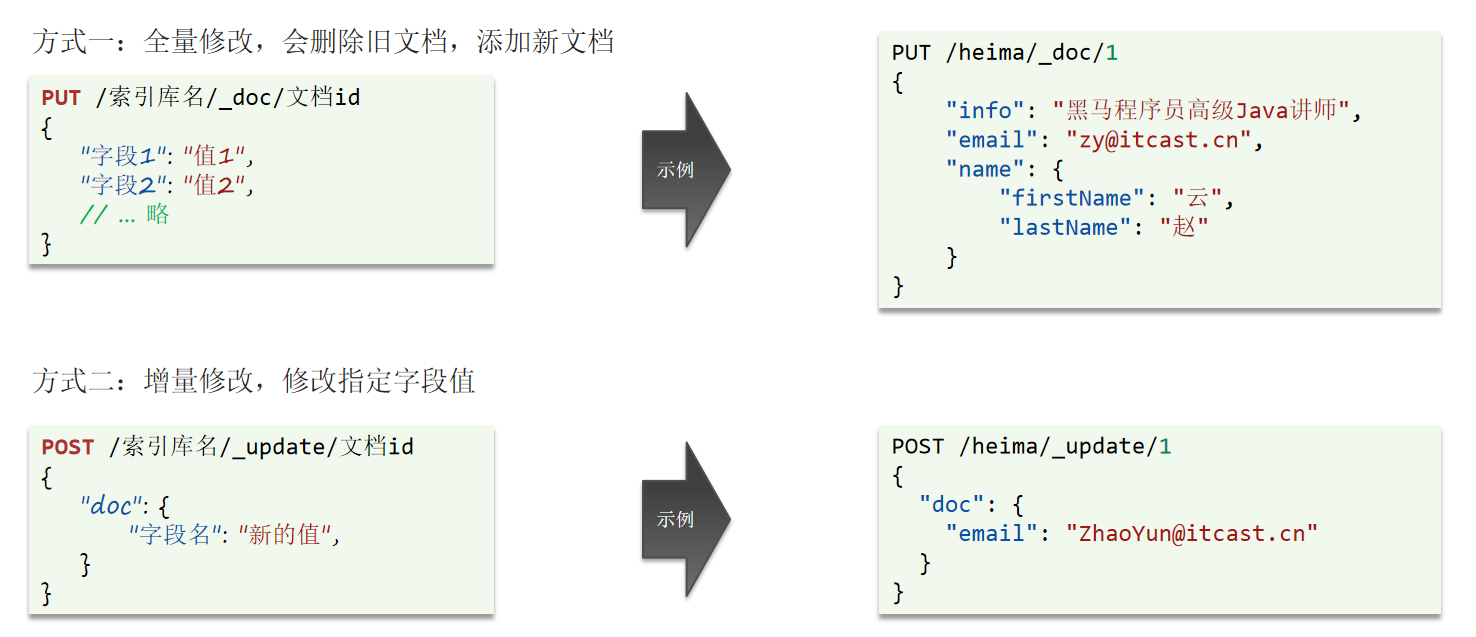
修改文档 两种方式

第一种 先全量修改 先依据文档id找到对应的文档给删除,在插入这样一条新的数据,如果数据库中没有这个文档id,那么就会新增这样一条数据,等同于增加操作

第二种 增量修改,修改指定字段POST /索引库名/\_update/文档id{

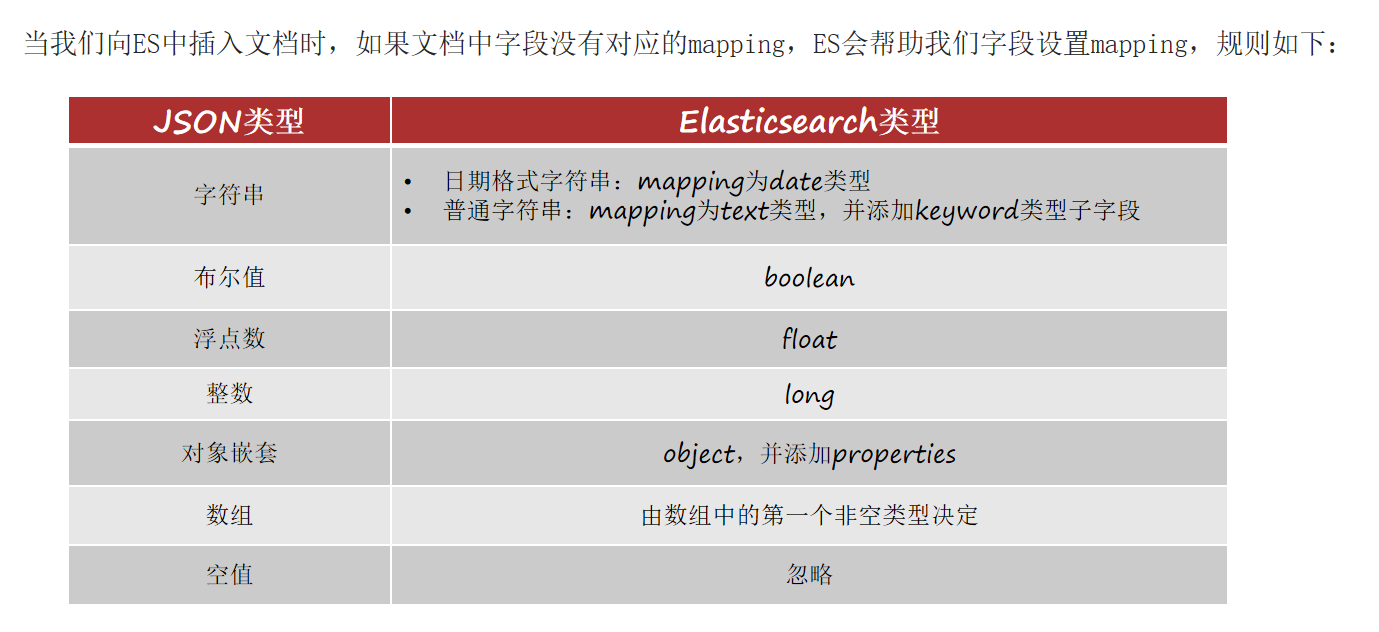
“字段名”:”新的值”,

.......}



插入文档时，es会检查文档中的字段是否有mapping，如果没有则按照默认mapping规则来创建索引。

如果默认mapping规则不符合你的需求，一定要自己设置字段mapping



1. Restclient操作数据库

ES官方提供了各种不同语言的客户端(比如说java语言)，用来操作ES。这些客户端的本质就是组装DSL语句，通过http请求发送给ES。官方文档地址：<https://www.elastic.co/guide/en/elasticsearch/client/index.html>

最早有java low level

后来出了 java high level 基于 low level封装 更加的方便和快捷

案例 :利用JavaRestClient实现创建、删除索引库，索引库操作的基本步骤：

初始化RestHighLevelClient

创建XxxIndexRequest。XXX是Create、Get、Delete

准备DSL（ Create时需要）

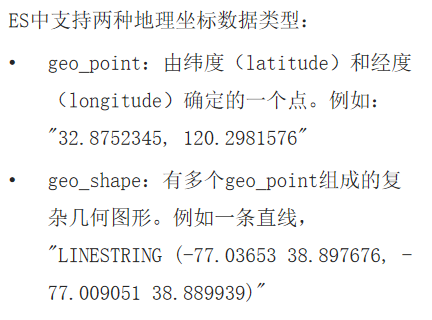
发送请求。调用RestHighLevelClient#indices().xxx()方法，xxx是create、exists、delete

1. 导入课前资料Demo,并运行sql文件 
2. 分析数据结构，定义mapping属性 字段名、数据类型、是否参与搜索、是否分词、如果分词，分词器是什么？
3. 初始化JavaRestClient
4. 利用JavaRestClient创建索引库
5. 利用JavaRestClient删除索引库
6. 利用JavaRestClient判断索引库是否存在

步骤二分析数据结构，定义mapping属性

Id这个字段比较特殊 必须是字符串 不分词 keyword

经纬度类型比较特殊 :geo\_point(一个点) 和geo\_shape(一个范围) 把经纬度两个字段合并到了一起虽然是字符串,但是类型为geo\_point



因为好多字段都要进行搜索条件是多个值而不是一个值,因为根据一个字段搜性能高,所以es给我们提供了一个属性copy\_to将当前字段拷贝到指定字段,示例如下比如说是all字段 并不是将文档拷贝进去了,而是基于他创建倒排索引,将来查的时候查不到这个字段,但是搜可以搜到



步骤三初始化JavaRestClient

1. 引入es的RestHighLevelClient依赖：

<dependency>  
 <groupId>org.elasticsearch.client</groupId>  
 <artifactId>elasticsearch-rest-high-level-client</artifactId>  
 </dependency>

1. 因为SpringBoot默认的ES版本是7.6.2，所以我们需要覆盖默认的ES版本：要保证版本一致

<properties>  
 <java.version>1.8</java.version>  
 <elasticsearch.version>7.12.1</elasticsearch.version>   
 </properties>



1. 初始化RestHighLevelClient：

RestHighLevelClient client = new RestHighLevelClient(RestClient.builder(  
 HttpHost.create("http://192.168.150.101:9200") es端口  
 ));



步骤四 利用javarestclient创建索引库

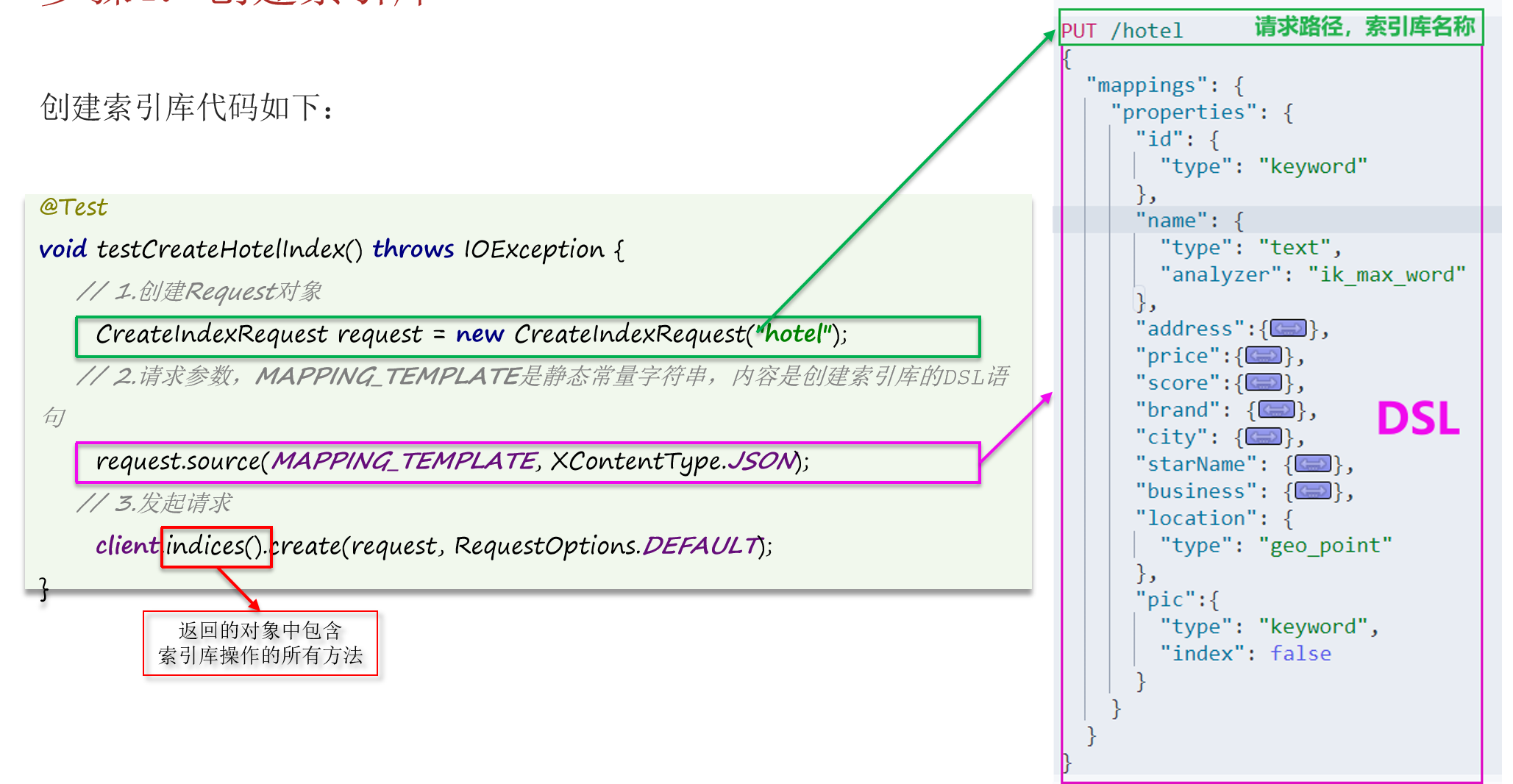
其实就是用java代码组织去发dsl语句

创建request对象 创建索引库请求CreateIndexRequset(指定索引库名)

请求参数source(dsl语句,请求风格XContentType.JSON(json))

发起请求client.indices(请求,请求头的参数一般都为默认) indices==index 代表的是索引库的所有操作增删改api indices就能拿到我们对索引库操作的所有方法 参数

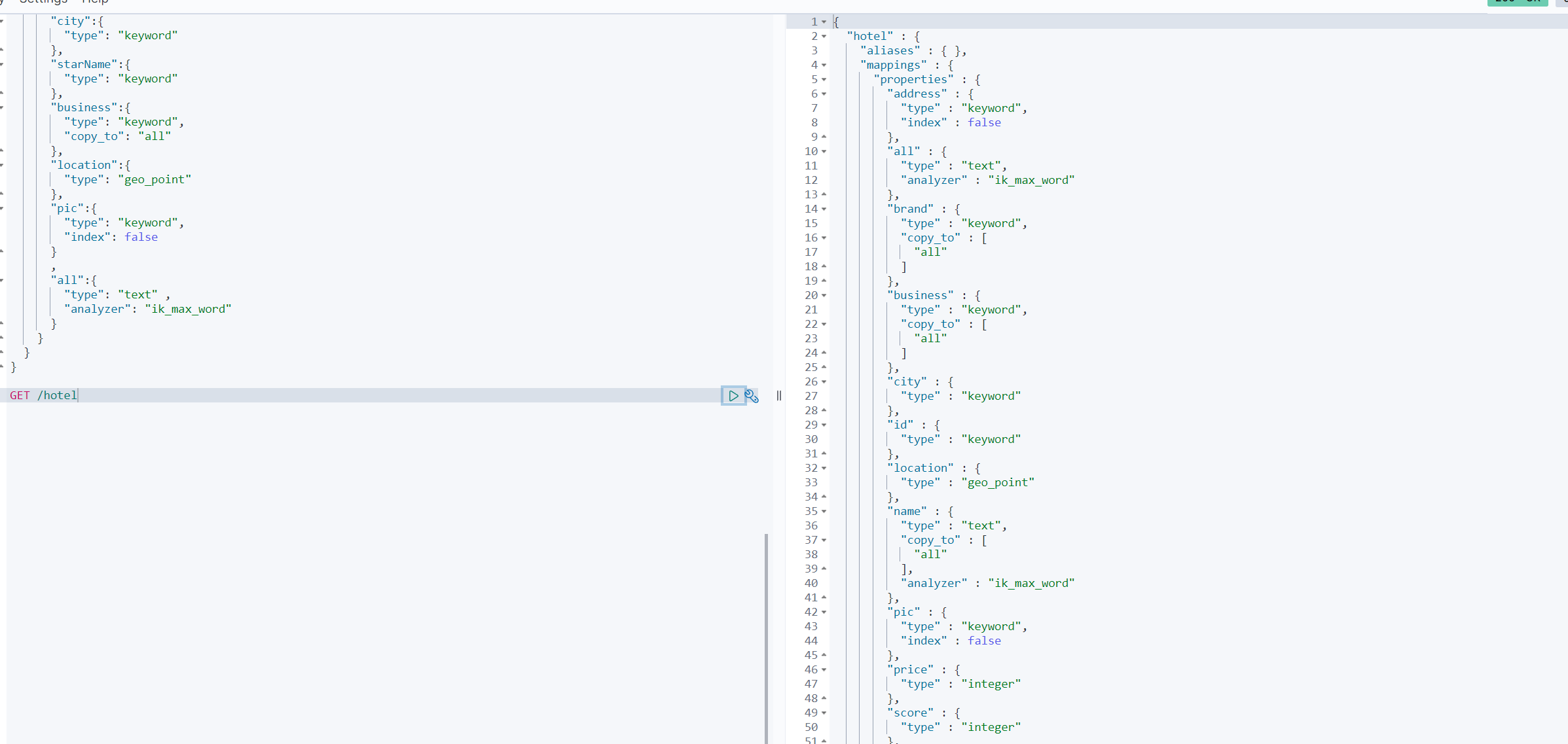
代码如下



Dsl语句就是步骤一分析出来的



执行后 在kibana执行 get /hotel 发现创建成功



删除索引库DeleteIndexRequest



判断索引库是否存在GetIndexRequest



1. restClient操作文档

文档操作的基本步骤：

初始化RestHighLevelClient

创建XxxRequest。XXX是Index、Get、Update、Delete

准备参数（Index和Update时需要）

发送请求。调用RestHighLevelClient#.xxx()方法，xxx是index、get、update、delete

解析结果（Get时需要）

去数据库查询酒店数据，导入到hotel索引库，实现酒店数据的CRUD。

基本步骤如下：

初始化JavaRestClient

利用JavaRestClient新增酒店数据 到索引库,到数据库查相关数据,保存到索引库,转换成对应的格式

利用JavaRestClient根据id查询酒店数据

利用JavaRestClient删除酒店数据

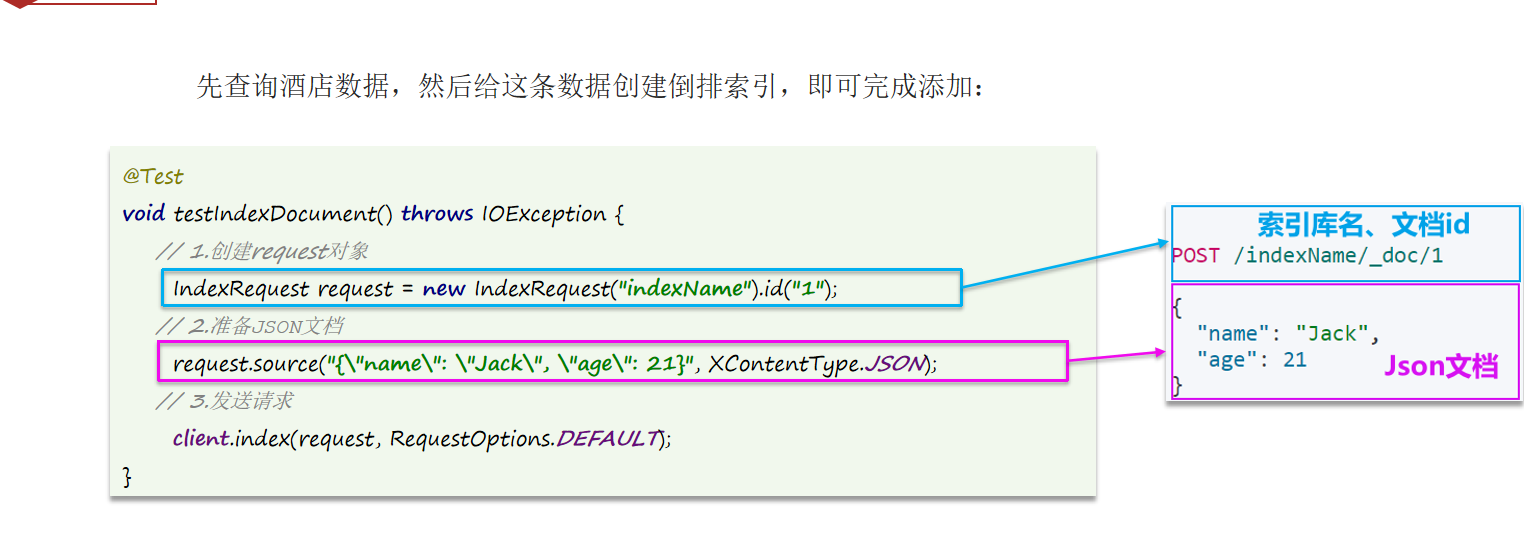
利用JavaRestClient修改酒店数据

步骤一 和对索引库操作一样:



步骤二: 向索引库中添加文档 指定哪个索引库给个id

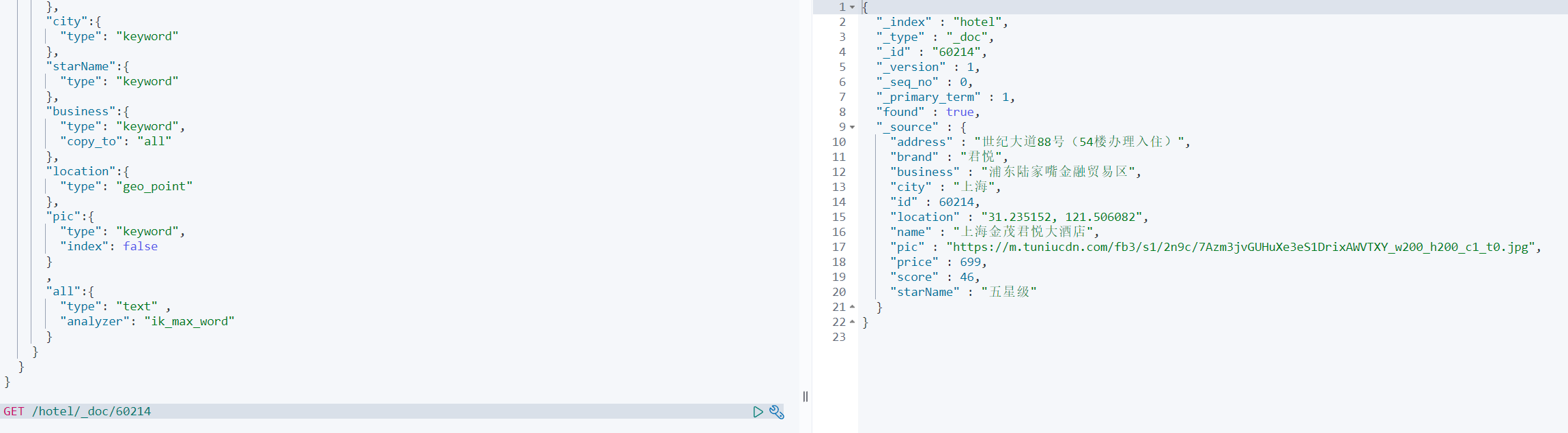
写dsl语句 默认 发送请求 index()相当于对文档操作



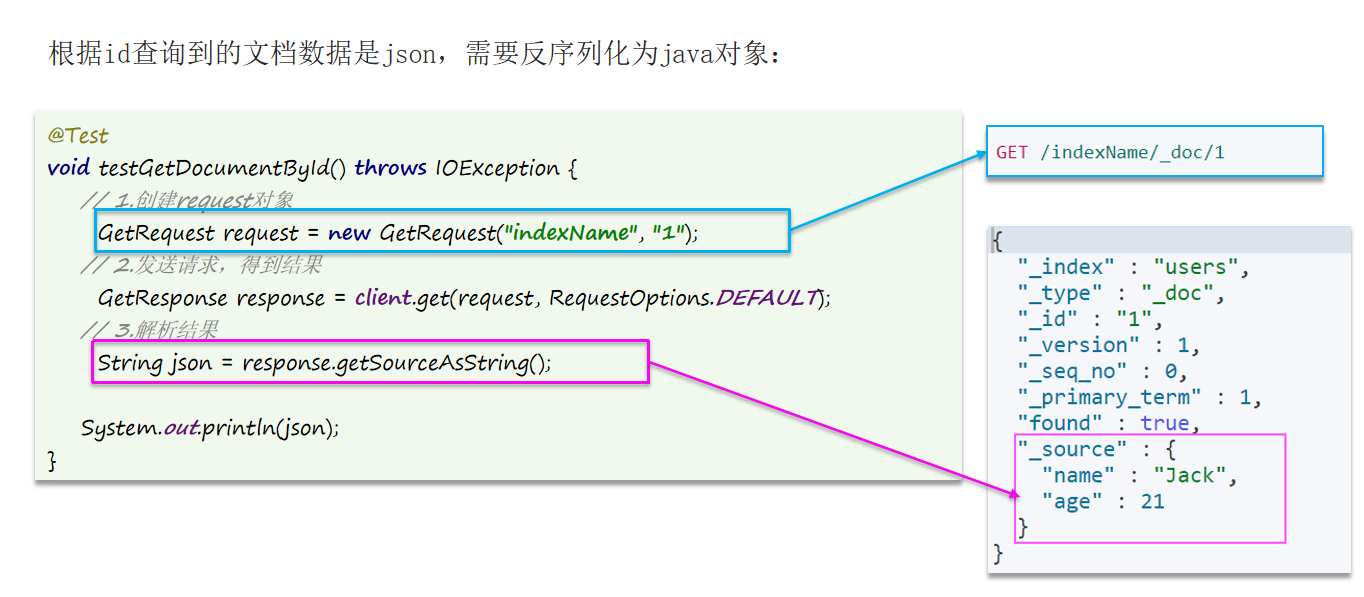
因为索引库的字段和我们数据库的字段不一样(经纬度 数据库是两个字段,索引库上就一个类型为geo\_point) 我们手动写了个hotelDoc构造方法对hotel进行转换 请求类型为IndexRequest



执行后 我们去kibana查询索引库为hotelid为60214就会是有数据的了 GET /hotel/\_doc/60214 经纬度拼接到了一起



步骤三: 根据id查询到文档数据 请求类型为GetRequest client.get(请求,默认) 然后得到sources 之后反序列化成我们的对象





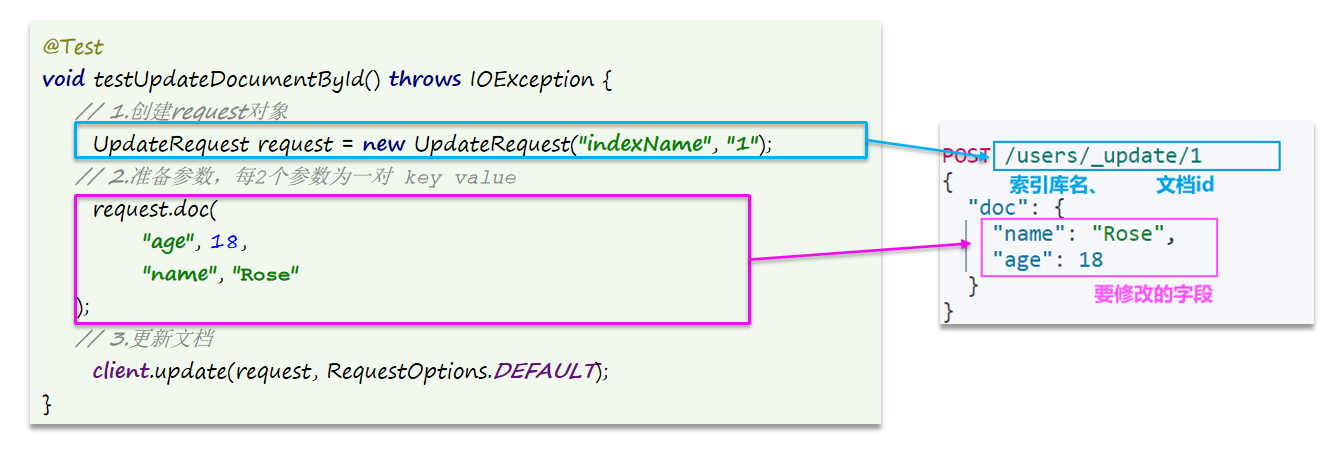
执行后,控制台成功打印出了文档转成对象的内容

步骤4：根据id修改酒店数据

修改文档数据有两种方式：

方式一：全量更新。再次写入id一样的文档，就会删除旧文档，添加新文档 和增加一样

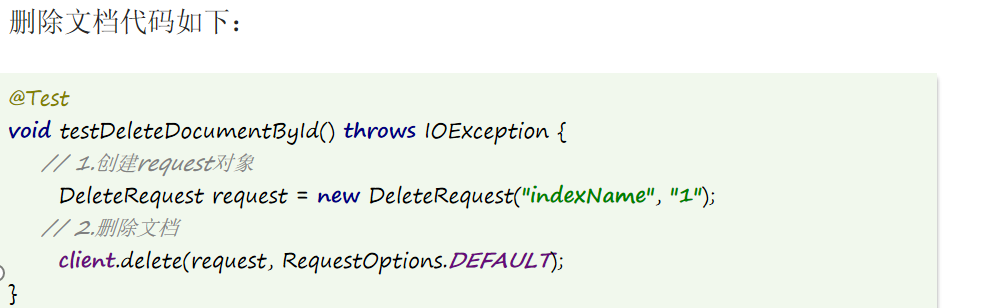
方式二：局部更新。只更新部分字段，我们演示方式二 请求类型为UpdateRequest client.update(request,RequestOptions.*DEFAULT*);





执行后,登录kibana查看这个id文档 发现数据修改了 GET /hotel/\_doc/60214

步骤5：根据id删除文档数据 请求类型为DeleteRequest client.delete(request,RequestOptions.*DEFAULT*);





利用JavaRestClient批量导入酒店数据到ES

思路：

利用mybatis-plus查询酒店数据

将查询到的酒店数据（Hotel）转换为文档类型数据（HotelDoc）

利用JavaRestClient中的Bulk批处理，实现批量新增文档，示例代码如下

请求类型bulkrequest client.bulk(request, RequestOptions.*DEFAULT*);

得到所有数据进行遍历 每遍历一条数据创建一个新增请求indexrequest 指定索引库,给id,将传值对象转换成JSON格式保存到bulkrequest对象中,遍历后提交bulkrequest请求 在kibana上看所有数据都保存到了 es中 GET /hotel/\_search

