[ElasticSearch 1](#_Toc8931)

[简介 1](#_Toc14178)

[什么是ElasticSearch 1](#_Toc25845)

[应用场景 1](#_Toc5245)

[单实例安装 2](#_Toc12496)

[插件安装 2](#_Toc8193)

[分布式安装 2](#_Toc30494)

[ElasticSearch基础概念 4](#_Toc28712)

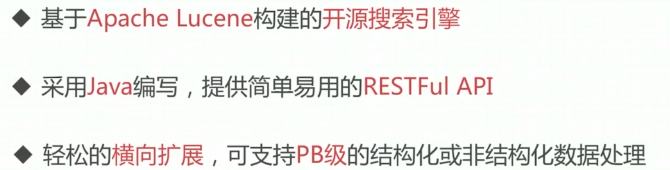
[ElasticSearch基本用法 5](#_Toc1181)

[创建结构化索引 9](#_Toc6259)

# ElasticSearch

## 简介

### 什么是ElasticSearch



### 应用场景



### 单实例安装

### 插件安装

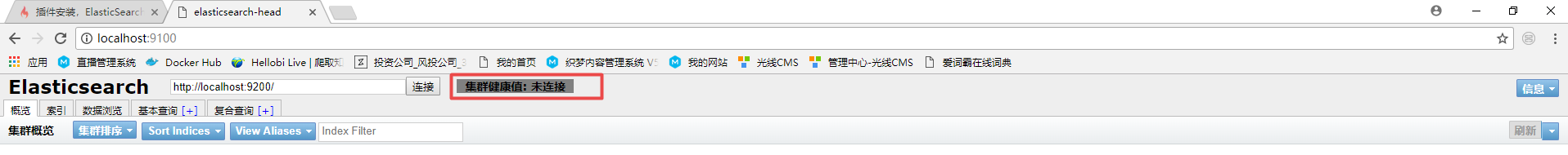
Head插件安装(用于ElasticSearch更好看的内容的NPM项目),使用npm install

<https://github.com/mobz/elasticsearch-head.git>

运行npm run start

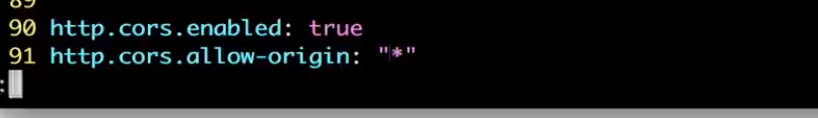
<http://localhost:9100>

首次加载的时候，我们会发现了未连接的现象，所以需要配置

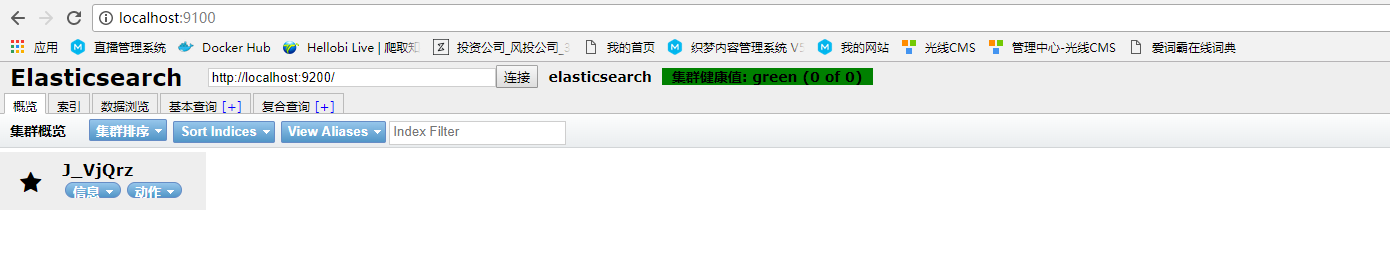


配置elasticsearch，解决上面的页面与ES数据间的页面跨域问题



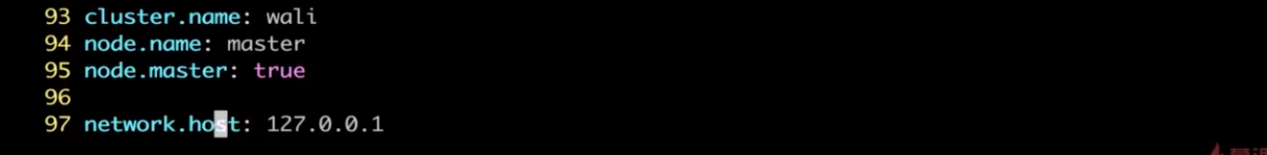


重新开发服务(server和head都要重新开启)

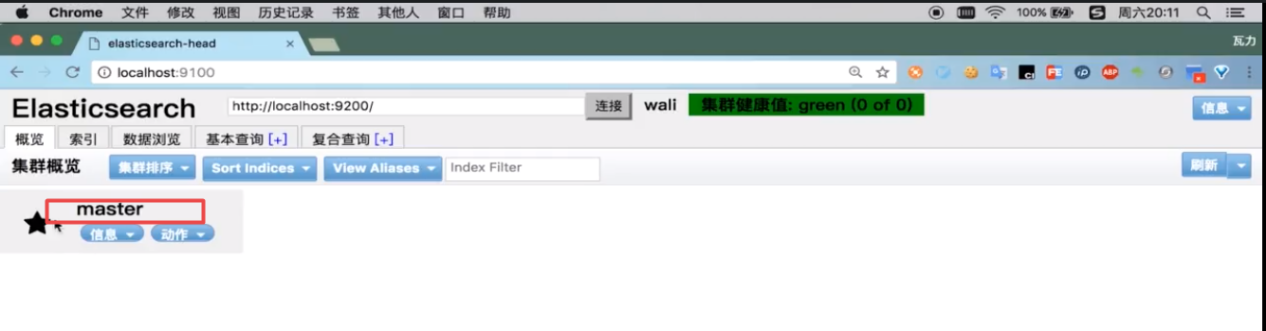


## 分布式安装

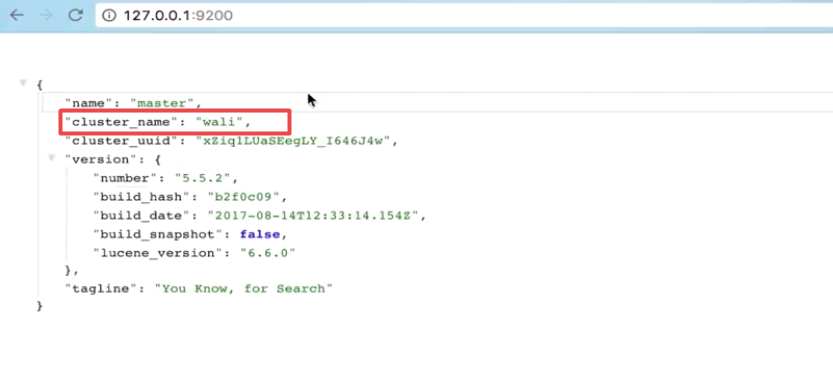
修改配置文件(添加完后，保存并退出，重启服务)



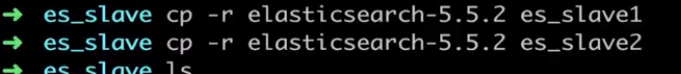
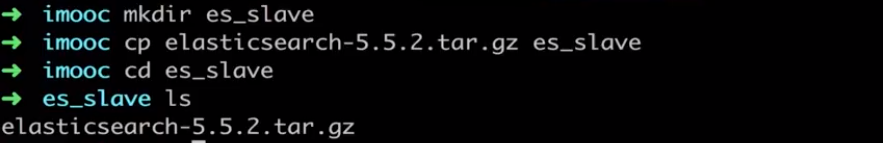
配置完后，网站的实例名称



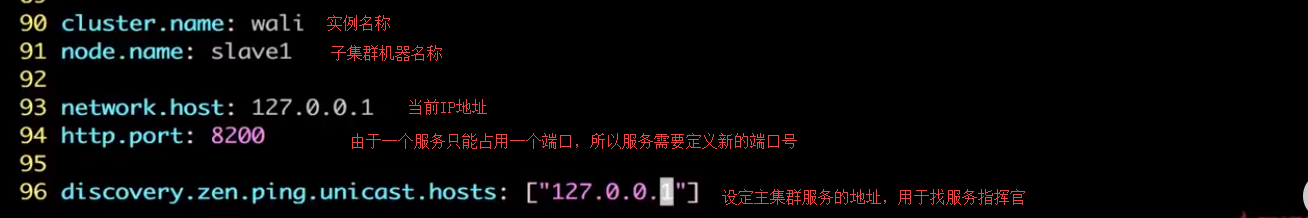
集群的名称

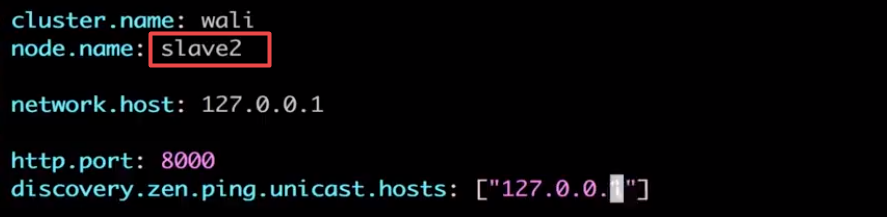


子集群构建：



子集群机的配置





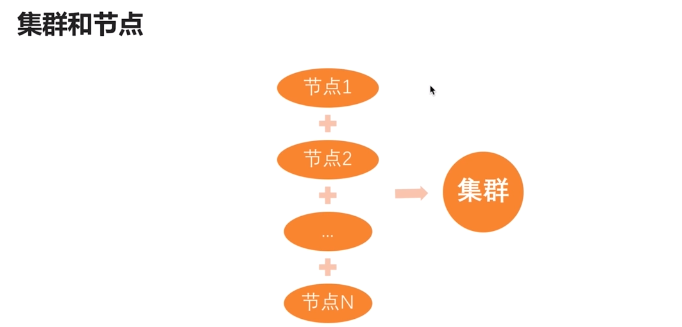
开启子集群机器

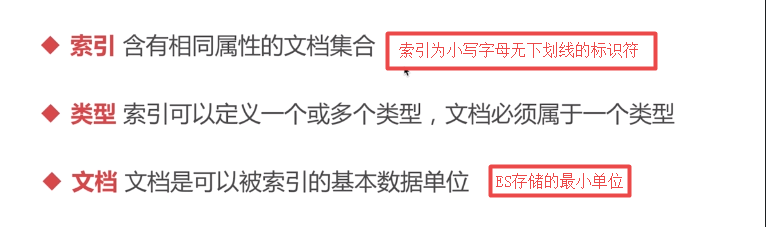


## ElasticSearch基础概念



两个重要的概念:即集群名称和子节点名称





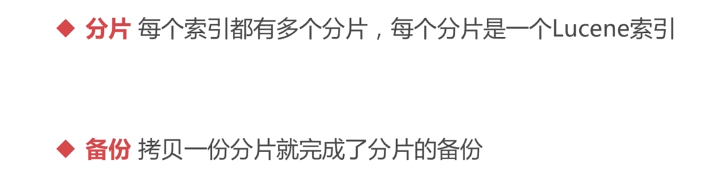
举一个具体的说法:

**索引相当于sql的数据库**

**类型相当于sql的table**

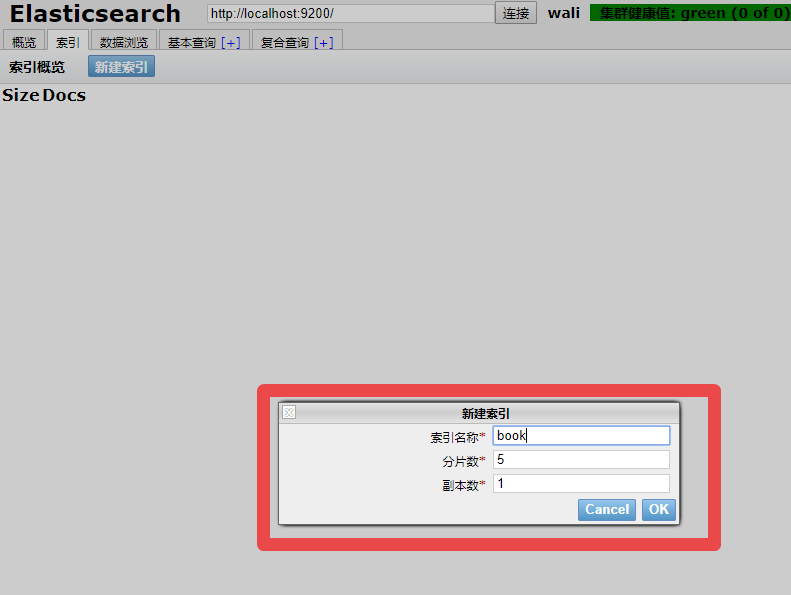
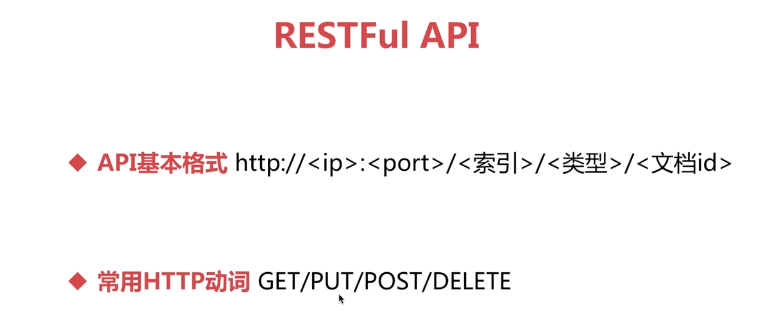
**文档相当于sql的记录**

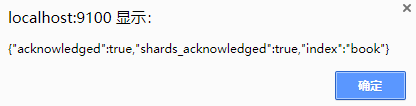
**ES默认会创建5个分片和一个备份，分片数量虽然可以修改，但只能在创建的时候进行修改，备份可以随时修改**



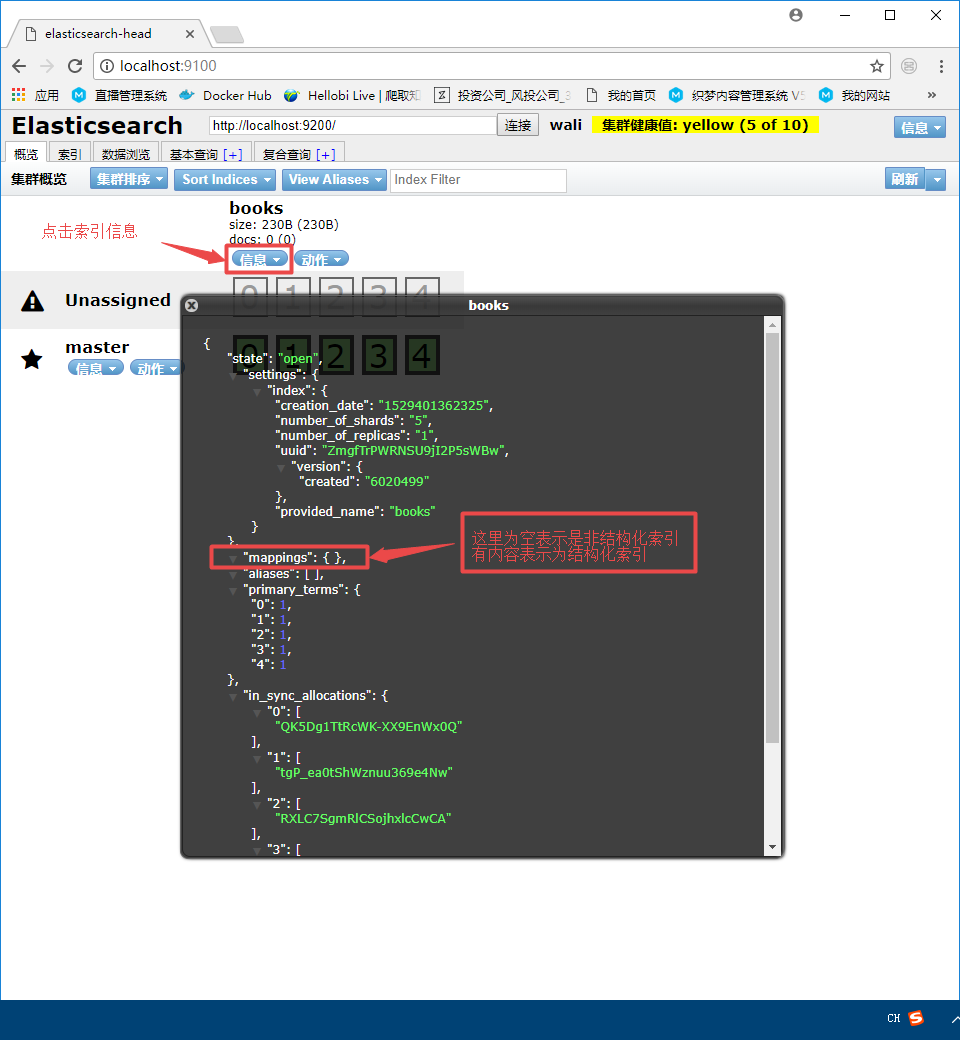
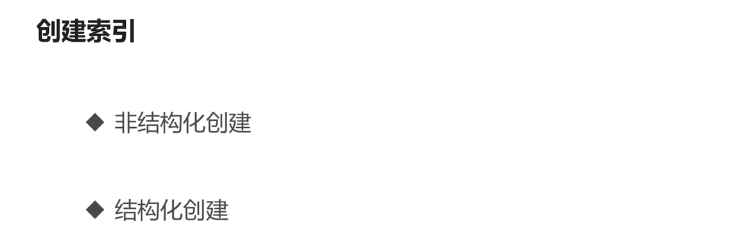
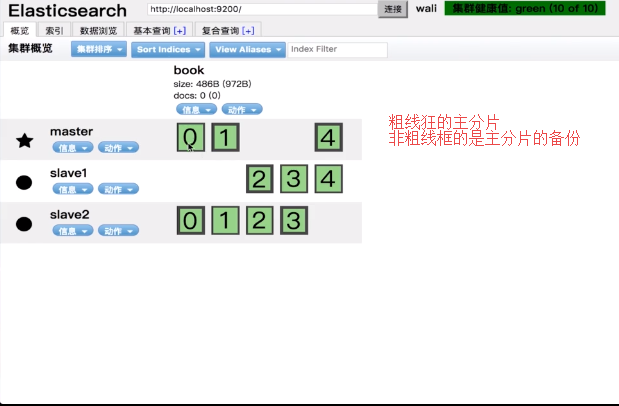
## ElasticSearch基本用法

访问ES的风格都是HTTP的Restful API





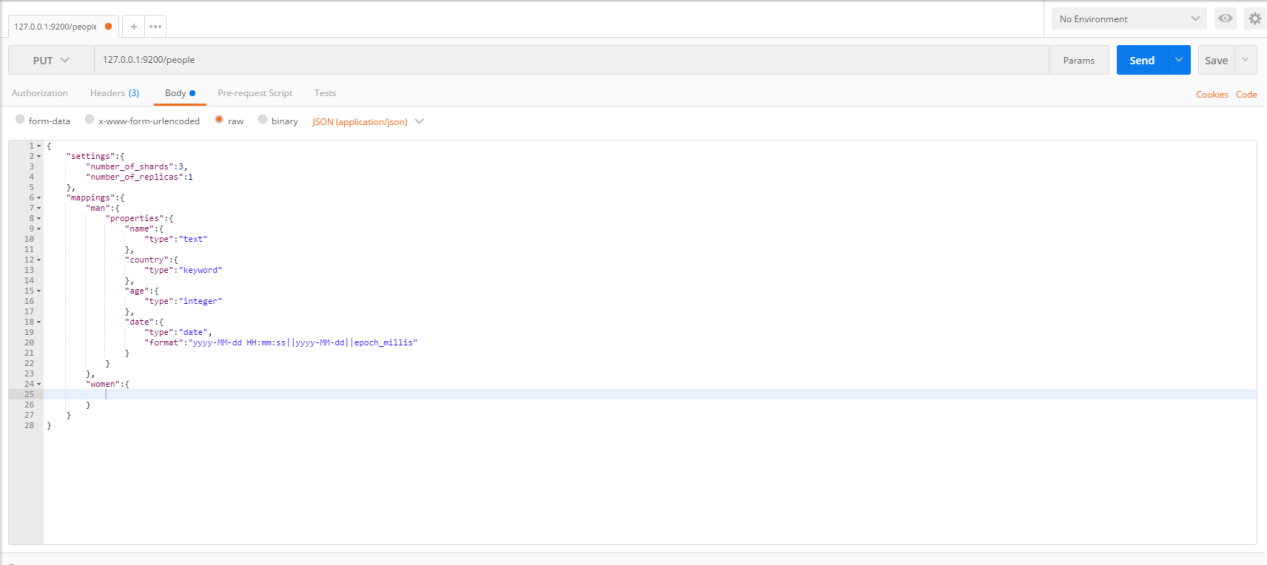
分片的备份数字对应的主分片的数字，一一对应



### 创建结构化索引



使用postman会方便一些



{

"settings":{

"number\_of\_shards":3,

"number\_of\_replicas":1

},

"mappings":{

"man":{

"properties":{

"name":{

"type":"text"

},

"country":{

"type":"keyword"

},

"age":{

"type":"integer"

},

"date":{

"type":"date",

"format":"yyyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch\_millis"

}

}

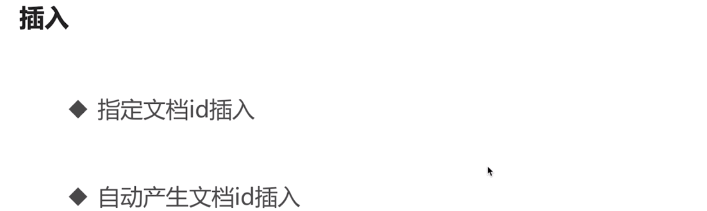
}

}

}



## 数据插入



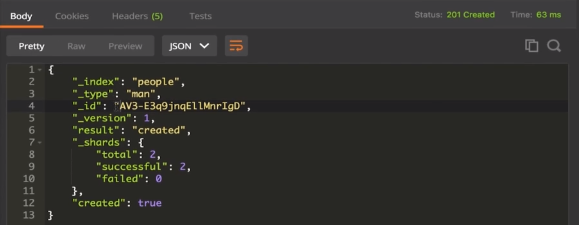
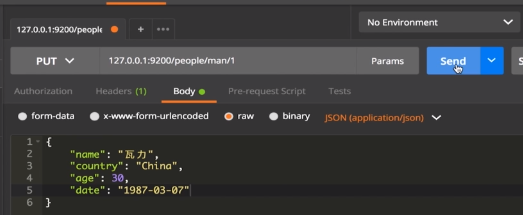
127.0.0.1:9200/people/man/1

people为索引

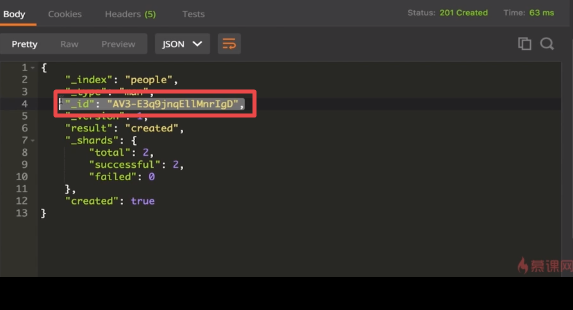
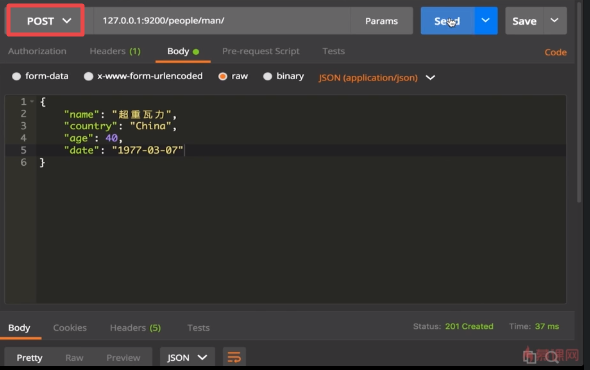
man 为类型

1 为ID

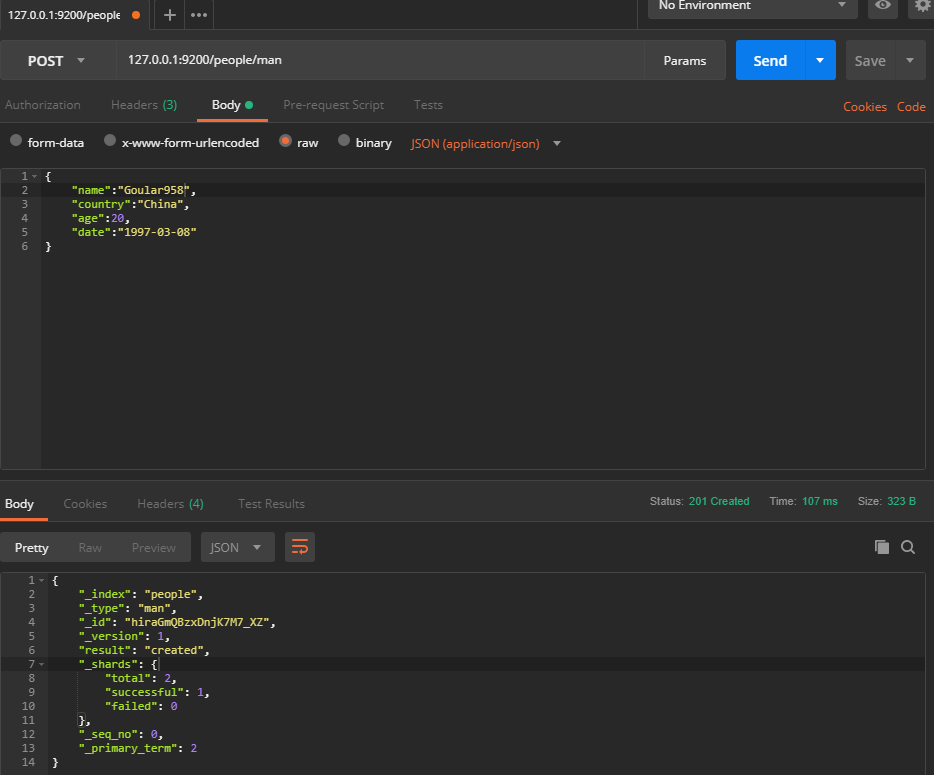
指定文档ID插入 -- 使用的是put方法会指定生成ID



自动产生文档ID插入 -- 使用post方法，其余与自定义id一样



自己的实践



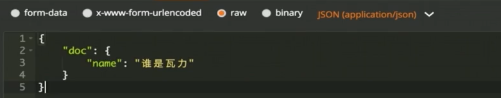
## 对文档进行修改 -- 即对数据行进行修改



例子:指定文档ID的修改



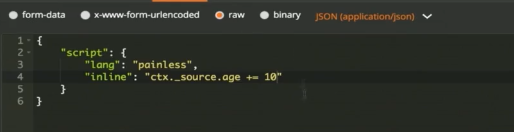
一定要添加doc属性



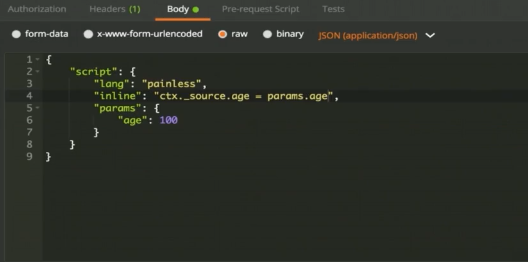
例子:通过脚本ID修改的文档

ES支持很多的脚本语言，painless为ES内置的脚本语言

写法1



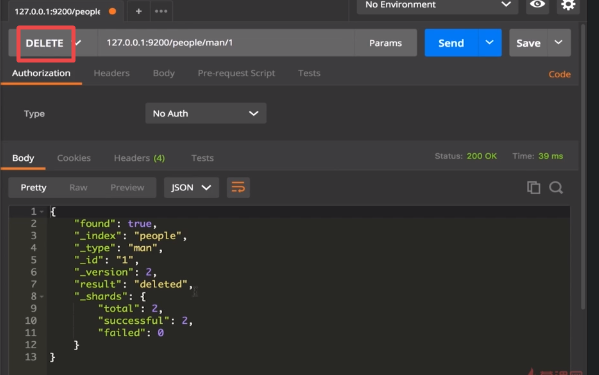
写法2



## 删除



删除文档的操作

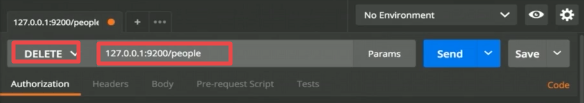


删除索引

方法一: 网页设定删除

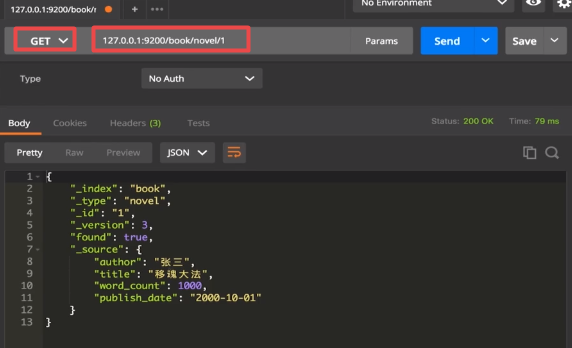


方法二:postman进行删除整个索引



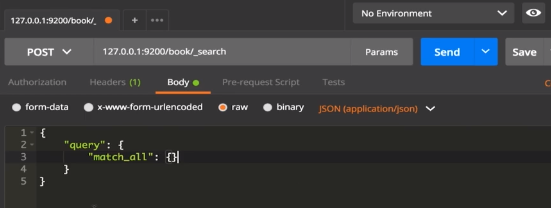
## 查询

根据ID查询内容

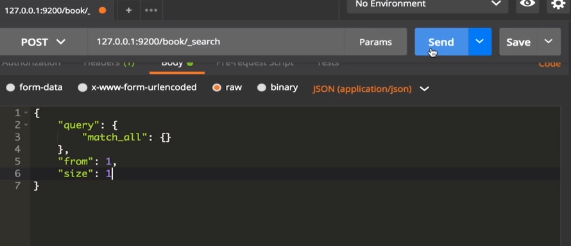


根据查询内容进行查询

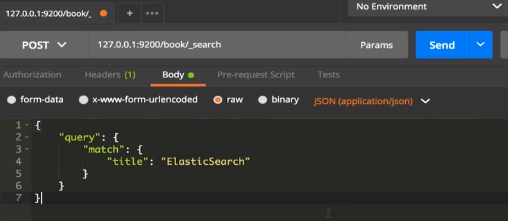
查询所有的数据



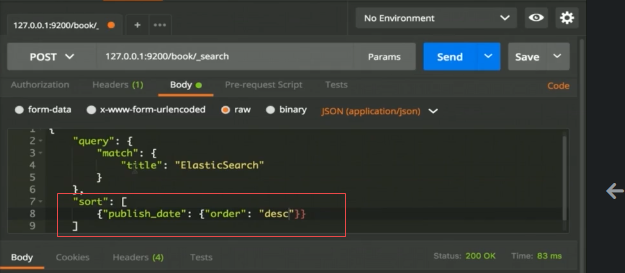
查询指定信息条数 ,当前查询的是一条的数据



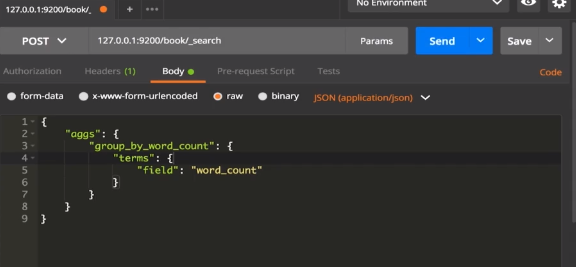
关键词查询

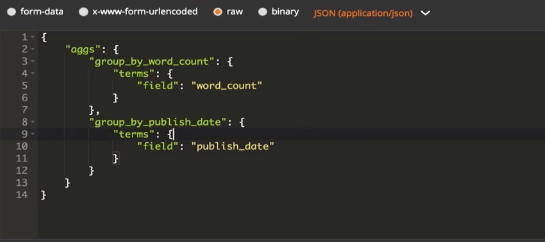
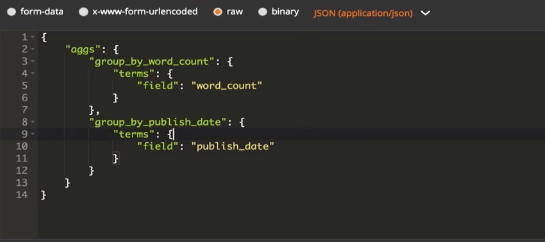


查询 -- 基于排序的查询

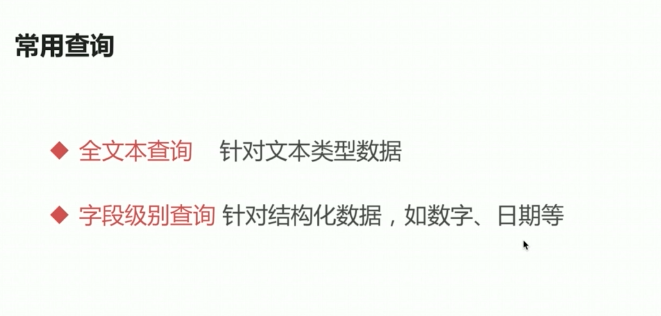
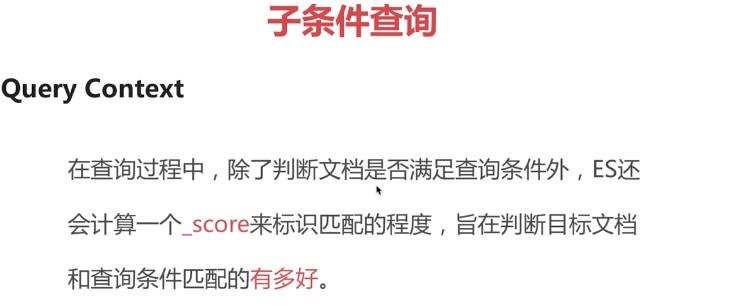
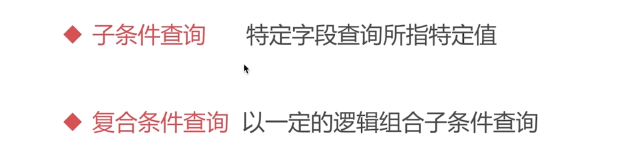


聚合查询

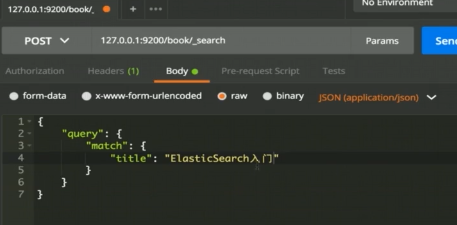




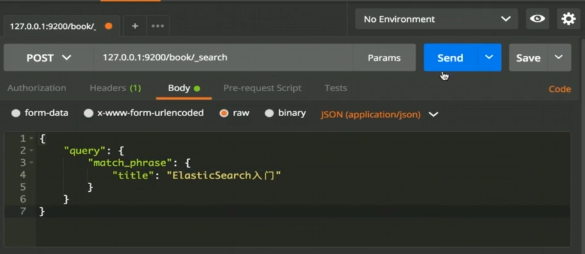
## ES高级查询



模糊匹配 -- 基本上所有的都会去查出来，但是依据score来判定内容，分数越高越匹配

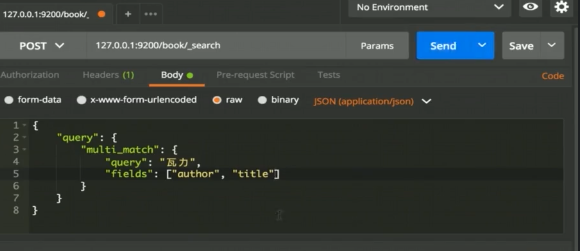


习语匹配 -- 即精确匹配

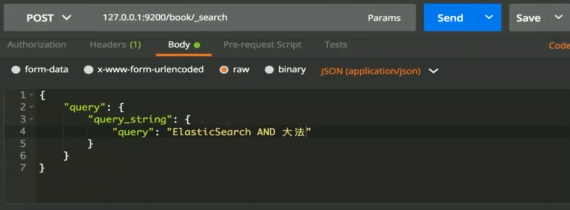


多个字段的模糊匹配查询

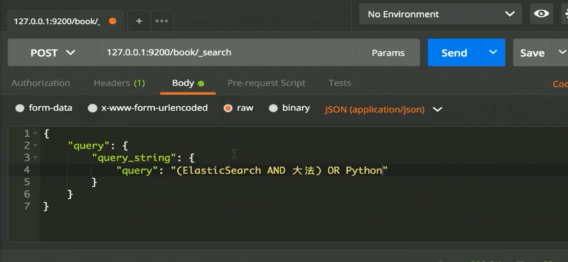
这里查询的是，只要autohr或者是title包含的查询内容都是可以进行数据获取的

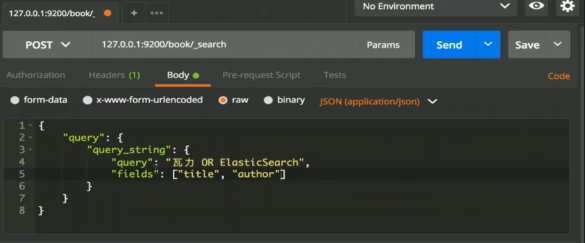


语法查询 --

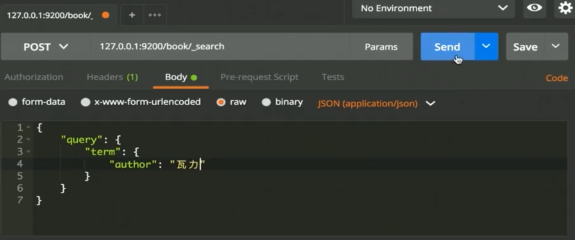


使用OR

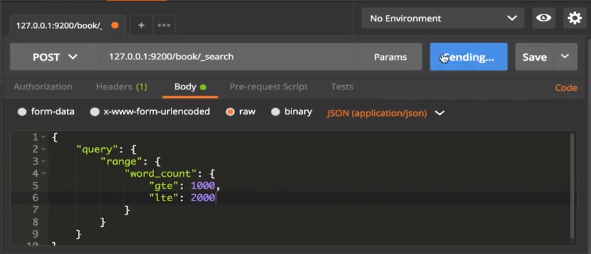


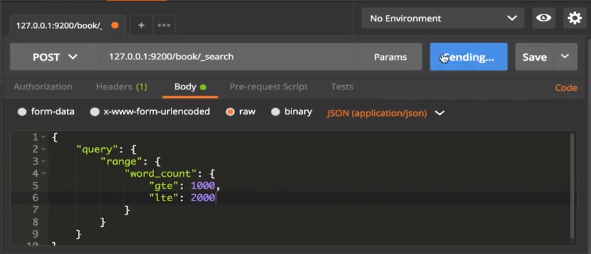


### 字段查询 -- 或者说是结构化的查询

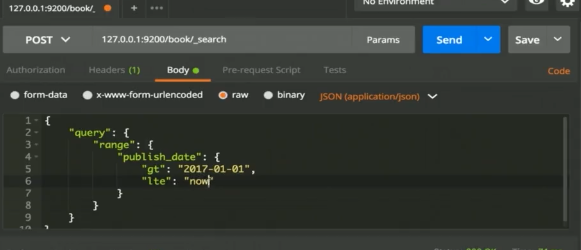


添加范围的的查询 -- range

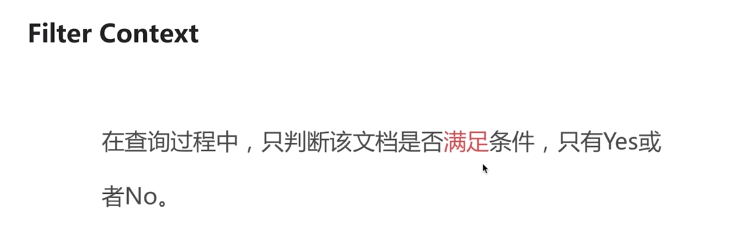


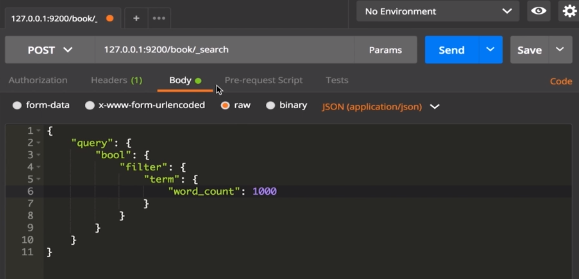


当前时间使用now

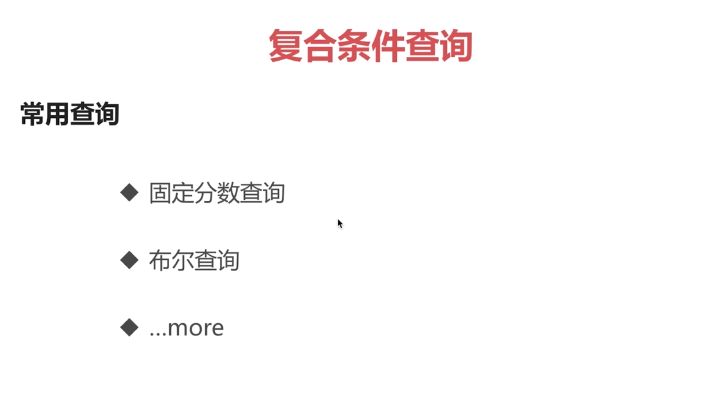


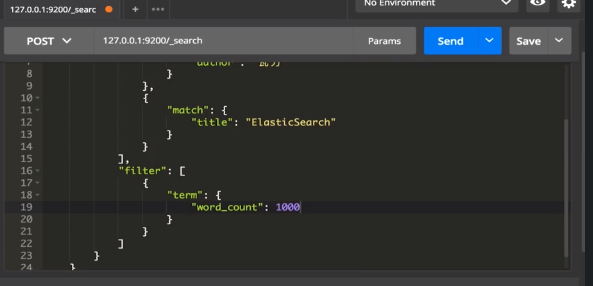
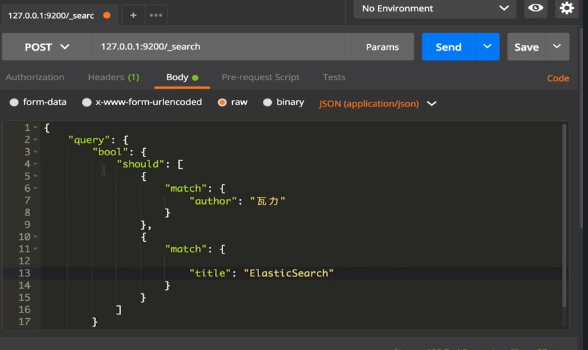
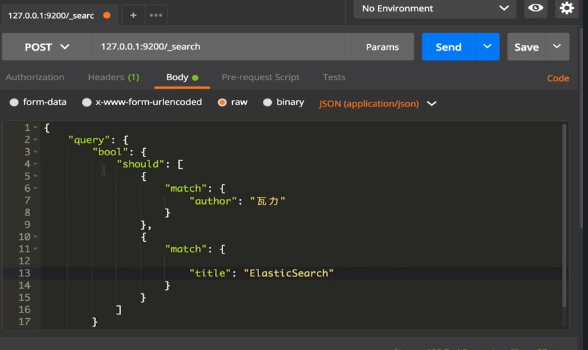
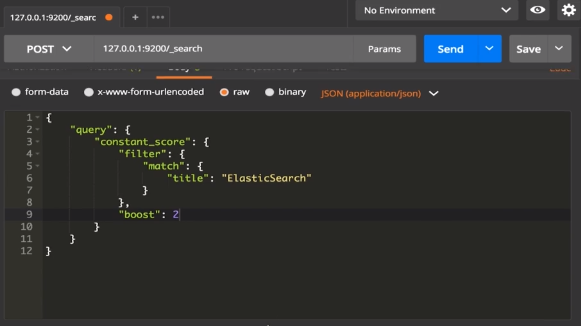
### 子条件查询 -- filter context



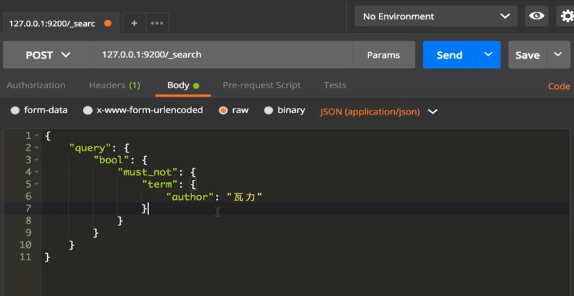


### 复合条件查询





必须不能是相关属性的内容作为查询的条件



## 实战演练 -- 课程使用了SpringBoot，这里我将使用golang来做

### 注意 -- ES的TCP端口为9300端口，HTTP端口为9200

创建es的访问实例

