ElasticSearch基本原理及分布式环境搭建

1、 ElasticSearch简介

Elaticsearch，简称为es， es是一个开源的高扩展的分布式全文检索引擎，它可以近乎实时的存储、检索数据；本 身扩展性很好，可以扩展到上百台服务器，处理PB级别的数据。es也使用Java开发并使用Lucene作为其核心来实 现所有索引和搜索的功能，但是它的目的是通过简单的RESTful API来隐藏Lucene的复杂性，从而让全文搜索变得 简单。

应用例子：

2013年初，GitHub抛弃了Solr，采取ElasticSearch 来做PB级的搜索。 “GitHub使用ElasticSearch搜索20TB 的数据，包括13亿文件和1300亿行代码”

维基百科：启动以elasticsearch为基础的核心搜索架构 SoundCloud：“SoundCloud使用ElasticSearch为1.8亿用户提供即时而精准的音乐搜索服务”

百度：百度目前广泛使用ElasticSearch作为文本数据分析，采集百度所有服务器上的各类指标数据及用户自 定义数据，通过对各种数据进行多维分析展示，辅助定位分析实例异常或业务层面异常。目前覆盖百度内部 20多个业务线（包括casio、云分析、网盟、预测、文库、直达号、钱包、风控等），单集群最大100台机 器，200个ES节点，每天导入30TB+数据

新浪使用ES 分析处理32亿条实时日志

阿里使用ES 构建挖掘自己的日志采集和分析体系

ElasticSearch对比Solr

- Solr 利用 Zookeeper 进行分布式管理，而 Elasticsearch 自身带有分布式协调管理功能;
- Solr 支持更多格式的数据，而 Elasticsearch 仅支持json文件格式;
- Solr 官方提供的功能更多，而 Elasticsearch 本身更注重于核心功能，高级功能多有第三方插件提供;
- Solr 在传统的搜索应用中表现好于 Elasticsearch，但在处理实时搜索应用时效率明显低于 Elasticsearch

2、ElasticSearch安装与启动

2.1 安装JDK(自行安装)

2.2 安装启动elasticsearch

```
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.8.0-x86_64.rpm
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.8.0-x86_64.rpm.sha512
shasum -a 512 -c elasticsearch-7.8.0-x86_64.rpm.sha512 #需要安装sha命令 yum install perl-Digest-SHA
rpm --install elasticsearch-7.8.0-x86_64.rpm

#安装完成后，可以在/etc/elasticsearch目录下看到相关的配置文件
#编辑 vim /etc/elasticsearch/elasticsearch.yml，增加以下两个配置，开启
# 是否支持跨域
http.cors.enabled: true
```

```
# *表示支持所有域名
http.cors.allow-origin: "*"
#放开如下配置（反正后面集群要用到）
node.name: node-1
network.host: 0.0.0.0 #DNS路由，提供外网访问，也可以为本机IP
cluster.initial_master_nodes: ["node-1"]

#配置开机启动
systemctl daemon-reload
systemctl enable elasticsearch.service

#如果不能用root启动，增加一个新账号
groupadd es
useradd es -g es -p es
chown -R es:es /es目录
su es

#启动
systemctl start elasticsearch
#测试是否安装成功
curl http://127.0.0.1:9200
#看到如下信息标识安装成功
[root@ydt elasticsearch]# curl http://127.0.0.1:9200
{
  "name" : "ydt",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "a-m51u-AQ6m-PaEJfwqTfg",
  "version" : {
    "number" : "7.8.0",
    "build_flavor" : "default",
    "build_type" : "rpm",
    "build_hash" : "757314695644ea9a1dc2fecd26d1a43856725e65",
    "build_date" : "2020-06-14T19:35:50.234439Z",
    "build_snapshot" : false,
    "lucene_version" : "8.5.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
#也可以通过命令查看启动状态
[root@ydt elasticsearch-head]# systemctl status elasticsearch
● elasticsearch.service - Elasticsearch
   Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; disabled;
vendor preset: disabled)
   Active: active (running) since 三 2020-07-08 17:09:00 CST; 16min ago
     Docs: https://www.elastic.co
 Main PID: 129764 (java)
    Tasks: 55
   CGroup: /system.slice/elasticsearch.service
           ├─129764 /usr/share/elasticsearch/jdk/bin/java -Xshare:auto -
Des.networkaddress.cache.ttl=60 -Des.networkaddress.cache.negative.ttl=10 -
XX:+AlwaysPreTouch -Xss1m -Djava.awt.headless=true -Dfil...
           └─129931 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-
x86_64/bin/controller

7月 08 17:08:31 ydt systemd[1]: Starting Elasticsearch...
7月 08 17:09:00 ydt systemd[1]: Started Elasticsearch.
```

```
#安装完es后通过 cut -d : -f 1 /etc/passwd 查看当前机器用户时可以发现 elasticearch 账户:
#7.8版本创建了elasticearch关联账号，关联了root，可以用root操作

#开放端口
firewall-cmd --zone=public --add-port=9100/tcp --permanent
firewall-cmd --zone=public --add-port=9200/tcp --permanent
firewall-cmd --zone=public --add-port=9300/tcp --permanent
#重启防火墙
firewall-cmd --reload
```

2.3 安装ES的图形化界面插件

elasticsearch-head是elasticsearch的集群管理工具，可以用于数据的浏览和查询。elasticsearch-head是一款开源软件，被托管在github上面，所以如果我们要使用它，必须先安装git,通过git获取elasticsearch-head,运行elasticsearch-head会用到grunt,而grunt需要npm包管理器，所以nodejs是必须要安装的。

安装git

```
yum install git
git --version #查看版本
```

安装nodeJS

```
yum install -y nodejs
node -v #查看版本
```

安装elasticsearch-head

```
git clone git://github.com/mobz/elasticsearch-head.git
cd elasticsearch-head
npm install
# 安装grunt命令
npm install -g grunt-cli
cd _site/
vim app.js #将localhost:9200修改为自己本地ip:9200，避免测试的时候手动去改，麻烦
#启动elasticsearch-head
grunt server
```

2.4 安装IK分词器

IK分词器是比较好的兼容中文的分词器，并且可以自定义字典，其他如庖丁解牛分词器也是不错的中文分词器

```
#进入elasticsearch安装目录
cd /usr/share/elasticsearch/plugins
mkdir analysis-ik
cd analysis-ik
#注意版本要与elasticsearch对应
wget https://github.com/medcl/elasticsearch-analysis-
ik/releases/download/v7.8.0/elasticsearch-analysis-ik-7.8.0.zip
unzip elasticsearch-analysis-ik-7.8.0.zip
#重启elasticsearch
systemctl restart elasticsearch
#测试校验
curl http://127.0.0.1:9200/_cat/plugins
```

最后面一直要看到如下图示标识启动成功：

3、ElasticSearch相关术语

3.1 ElasticSearch与关系型数据库的对比

3.1.1 相关性

Elasticsearch是面向文档(document oriented)的，这意味着它可以存储整个对象或文档(document)。然而它不仅仅是存储，还会索引(index)每个文档的内容使之可以被搜索。在Elasticsearch中，你可以对文档（而非成行成列的 数据）进行索引、搜索、排序、过滤。Elasticsearch比传统关系型数据库如下：

```
Relational DB -> Databases -> Tables -> Rows -> Columns
Elasticsearch -> Indices -> Types -> Documents -> Fields
```

3.1.2 优劣

数据库查询存在的问题：

性能低：使用模糊查询，左边有通配符，不会走索引，会全表扫描，性能低

功能弱：因为查询条件不能拆分词条，比如"华为5G手机",使用"华为手机"作为条件查询不出来数据

Elasticsearch存在的问题

- MySQL有事务性,而ElasticSearch没有事务性,所以你删了的数据是无法恢复的。
- ElasticSearch没有物理外键这个特性，,如果你的数据强一致性要求比较高,还是建议慎用

PS:ElasticSearch和MySql分工不同，MySQL负责存储数据，ElasticSearch负责搜索数据。

3.2 Elasticsearch核心概念

3.2.1 **索引 index**

一个索引就是一个拥有几分相似特征的文档的集合。比如说，你可以有一个客户数据的索引，另一个产品目录的索 引，还有一个订单数据的索引。一个索引由一个名字来标识（必须全部是小写字母的），并且当我们要对对应于这 个索引中的文档进行索引、搜索、更新和删除的时候，都要使用到这个名字。在一个集群中，可以定义任意多的索 引。

3.2.2 类型 type

在一个索引中，你可以定义一种或多种类型。一个类型是你的索引的一个逻辑上的分类/分区，其语义完全由你来 定。通常，会为具有一组共同字段的文档定义一个类型。比如说，我们假设你运营一个博客平台并且将你所有的数据存储到一个索引中。在这个索引中，你可以为用户数据定义一个类型，为博客数据定义另一个类型，当然，也可以为评论数据定义另一个类型。

3.2.3 字段**Field****

相当于是数据表的字段，对文档数据根据不同属性进行的分类标识

3.2.4 映射 mapping

mapping是处理数据的方式和规则方面做一些限制，如某个字段的数据类型、默认值、分析器、是否被索引等等，这些都是映射里面可以设置的，其它就是处理es里面数据的一些使用规则设置也叫做映射，按着最优规则处理数据 对性能提高很大，因此才需要建立映射，并且需要思考如何建立映射才能对性能更好。

3.2.5 文档 document

一个文档是一个可被索引的基础信息单元。比如，你可以拥有某一个客户的文档，某一个产品的一个文档，当然， 也可以拥有某个订单的一个文档。文档以JSON（Javascript Object Notation）格式来表示，而JSON是一个到处存 在的互联网数据交互格式。 在一个index/type里面，你可以存储任意多的文档。注意，尽管一个文档，物理上存在于一个索引之中，文档必须 被索引/赋予一个索引的type。

3.2.6 接近实时 NRT

Elasticsearch是一个接近实时的搜索平台。这意味着，从索引一个文档直到这个文档能够被搜索到有一个轻微的延迟（通常是1秒以内）

3.2.7 集群 cluster

Relational DB -> Databases -> Tables -> Rows -> Columns

Elasticsearch -> Indices -> Types -> Documents -> Fields

一个集群就是由一个或多个节点组织在一起，它们共同持有整个的数据，并一起提供索引和搜索功能。一个集群由 一个唯一的名字标识，这个名字默认就是“elasticsearch”。这个名字是重要的，因为一个节点只能通过指定某个集 群的名字，来加入这个集群

3.2.8 节点 node

一个节点是集群中的一个服务器，作为集群的一部分，它存储数据，参与集群的索引和搜索功能。和集群类似，一 个节点也是由一个名字来标识的，默认情况下，这个名字是一个随机的漫威漫画角色的名字，这个名字会在启动的 时候赋予节点。这个名字对于管理工作来说挺重要的，因为在这个管理过程中，你会去确定网络中的哪些服务器对 应于Elasticsearch集群中的哪些节点。 一个节点可以通过配置集群名称的方式来加入一个指定的集群。默认情况下，每个节点都会被安排加入到一个叫 做“elasticsearch”的集群中，这意味着，如果你在你的网络中启动了若干个节点，并假定它们能够相互发现彼此， 它们将会自动地形成并加入到一个叫做“elasticsearch”的集群中。 在一个集群里，只要你想，

可以拥有任意多个节点。而且，如果当前你的网络中没有运行任何Elasticsearch节点， 这时启动一个节点，会默认创建并加入一个叫做“elasticsearch”的集群。

3.2.9 分片和复制 shards&replicas

一个索引可以存储超出单个结点硬件限制的大量数据。比如，一个具有10亿文档的索引占据1TB的磁盘空间，而任 一节点都没有这样大的磁盘空间；或者单个节点处理搜索请求，响应太慢。为了解决这个问题，Elasticsearch提供 了将索引划分成多份的能力，这些份就叫做分片。当你创建一个索引的时候，你可以指定你想要的分片的数量。每 个分片本身也是一个功能完善并且独立的“索引”，这个“索引”可以被放置到集群中的任何节点上。分片很重要，主 要有两方面的原因： 1）允许你水平分割/扩展你的内容容量。 2）允许你在分片（潜在地，位于多个节点上）之上 进行分布式的、并行的操作，进而提高性能/吞吐量。 至于一个分片怎样分布，它的文档怎样聚合回搜索请求，是完全由Elasticsearch管理的，对于作为用户的你来说， 这些都是透明的。 在一个网络/云的环境里，失败随时都可能发生，在某个分片/节点不知怎么的就处于离线状态，或者由于任何原因 消失了，这种情况下，有一个故障转移机制是非常有用并且是强烈推荐的。为此目的，Elasticsearch允许你创建分 片的一份或多份拷贝，这些拷贝叫做复制分片，或者直接叫复制。 复制之所以重要，有两个主要原因： 在分片/节点失败的情况下，提供了高可用性。因为这个原因，注意到复制分 片从不与原/主要（original/primary）分片置于同一节点上是非常重要的。扩展你的搜索量/吞吐量，因为搜索可以 在所有的复制上并行运行。总之，每个索引可以被分成多个分片。一个索引也可以被复制0次（意思是没有复制） 或多次。一旦复制了，每个索引就有了主分片（作为复制源的原来的分片）和复制分片（主分片的拷贝）之别。分 片和复制的数量可以在索引创建的时候指定。在索引创建之后，你可以在任何时候动态地改变复制的数量，但是你 事后不能改变分片的数量。 默认情况下，Elasticsearch中的每个索引被分片5个主分片和1个复制，这意味着，如果你的集群中至少有两个节 点，你的索引将会有5个主分片和另外5个复制分片（1个完全拷贝），这样的话每个索引总共就有10个分片。

4、ElasticSearch的客户端操作

实际开发中，主要有三种方式可以作为elasticsearch服务的客户端：

第一种，elasticsearch-head插件

第二种，使用elasticsearch提供的Restful接口直接访问（Postman）

第三种，使用elasticsearch提供的API进行访问

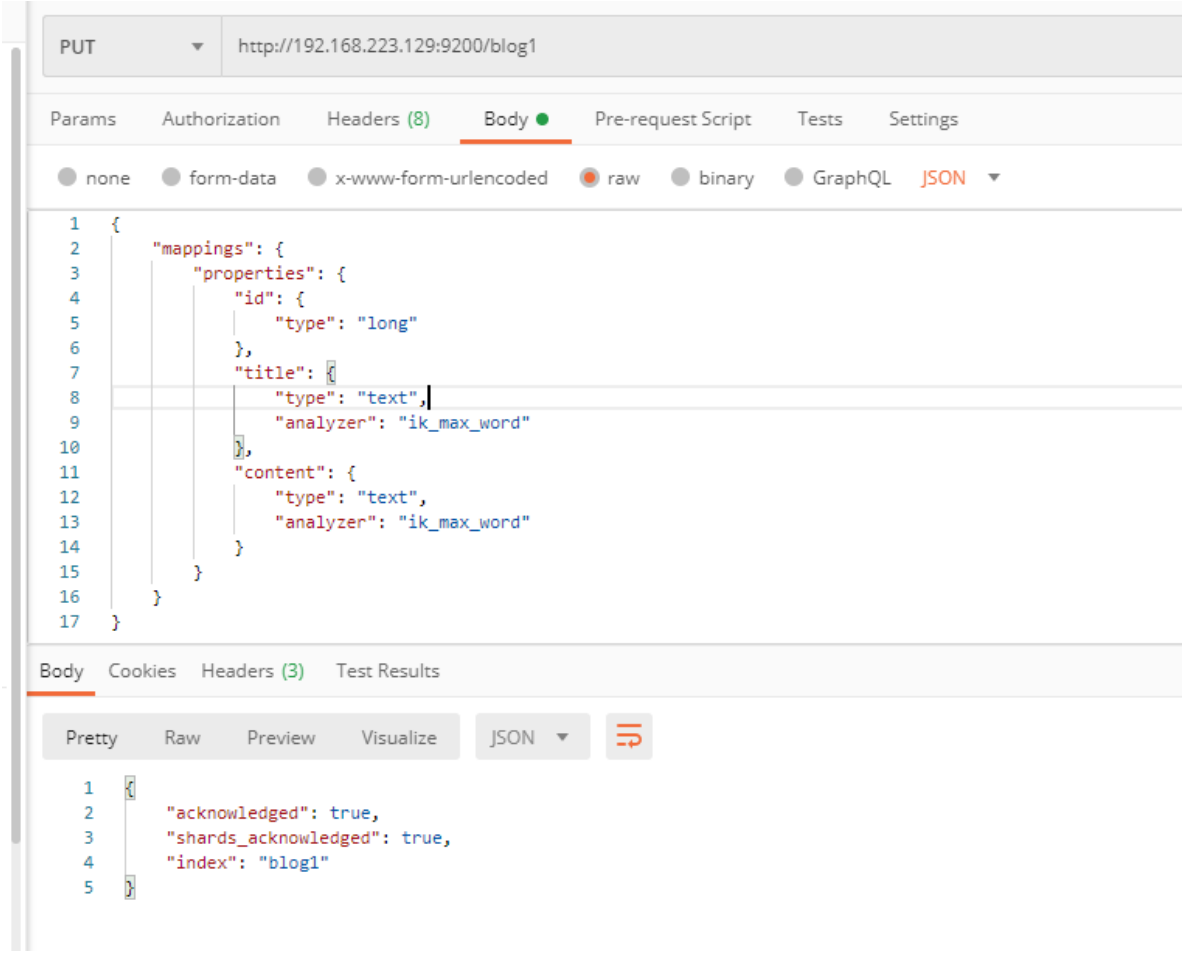
PS：各版本的ES，API接口可能不太一样，本课程以ES7以后的作为依据！

4.1 使用Postman工具进行Restfu接口访问

4.1.1 创建索引index和映射mapping

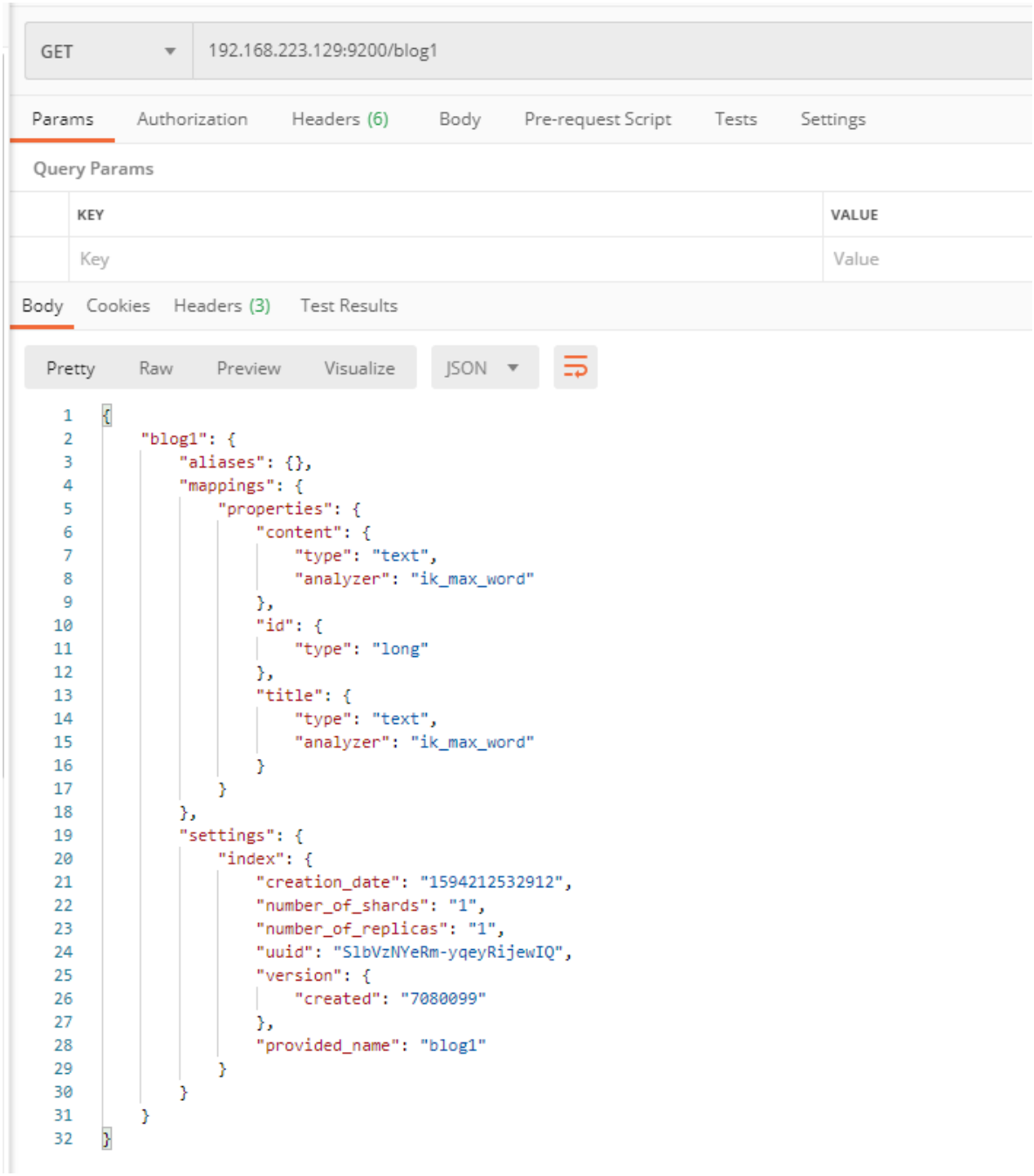
```
请求url：
PUT http://192.168.223.129:9200/blog1
请求体：
{
    "mappings": {
        "properties": {
            "id": {
                "type": "long"
            },
            "title": {
                "type": "text",
                "analyzer": "ik_max_word"
            },
            "content": {
```

```
                "type": "text",
                "analyzer": "ik_max_word"
            }
        }
    }
}
```



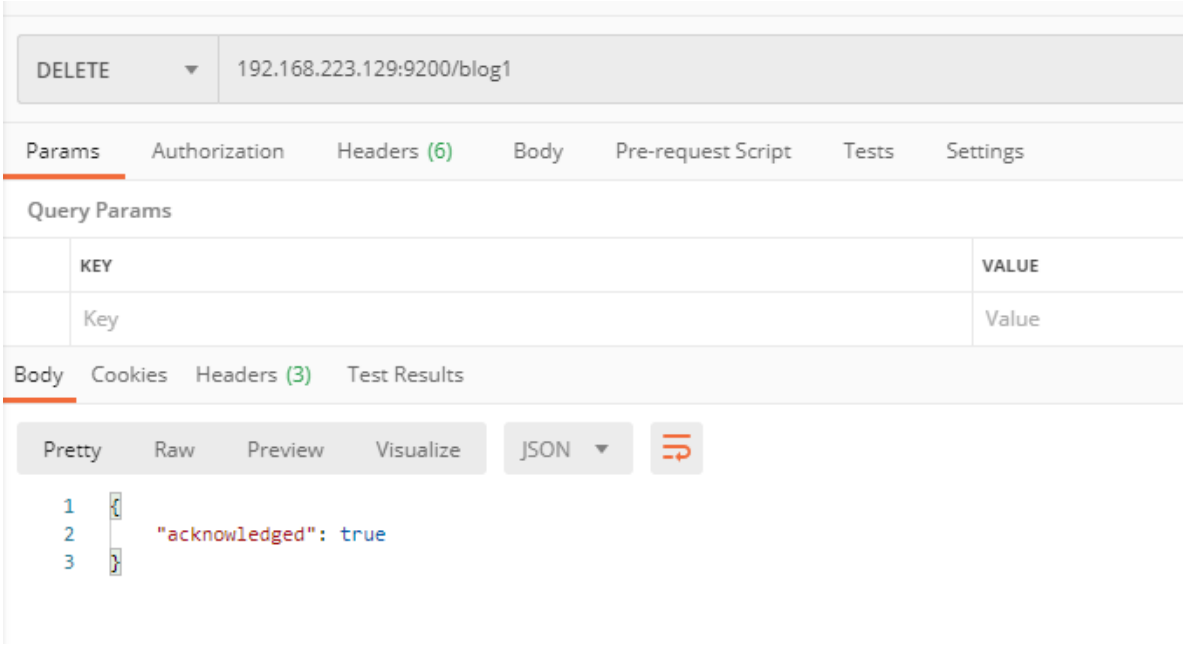
4.1.2 获取索引index

```
GET http://192.168.223.129:9200/blog1
```



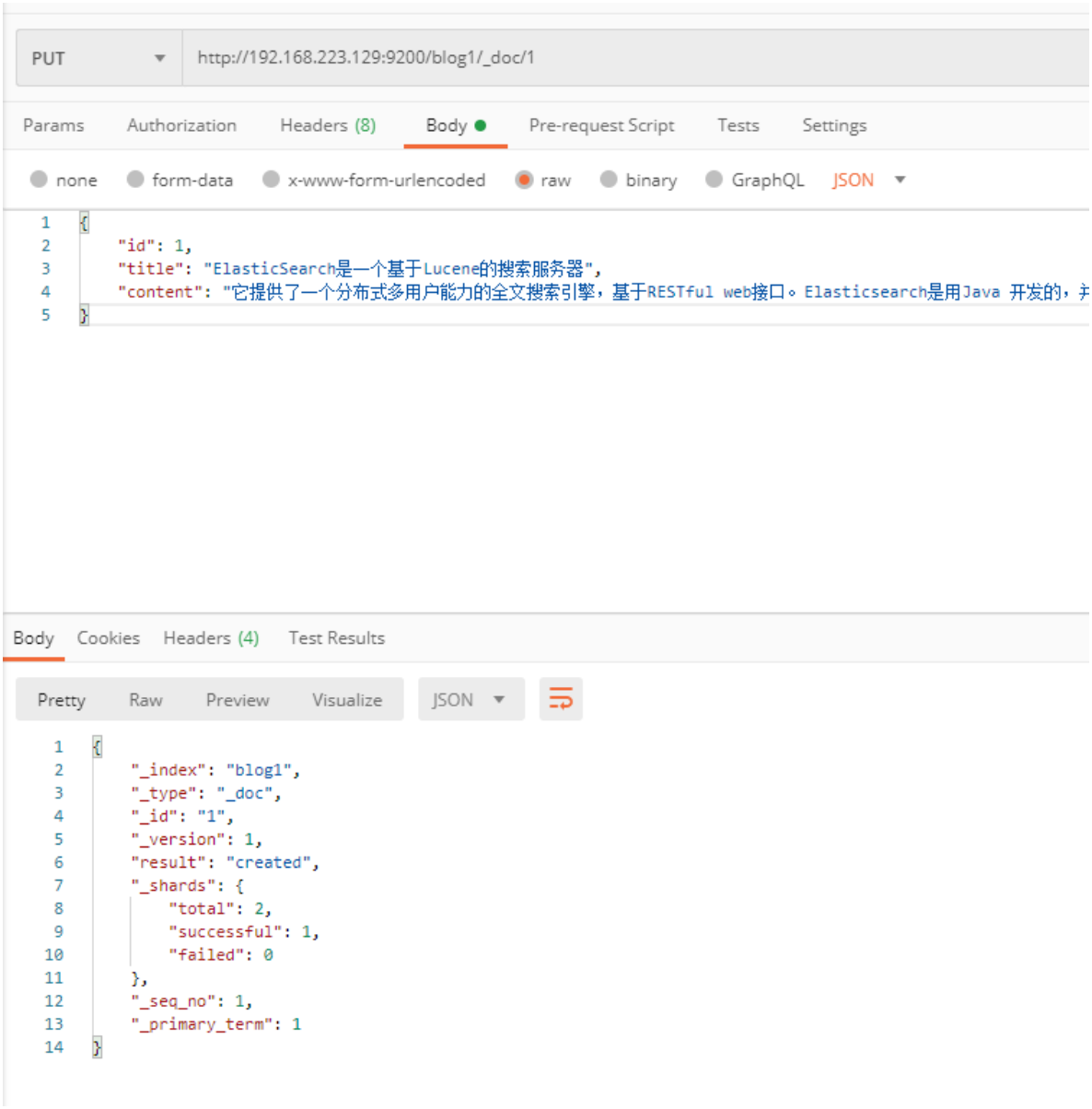
4.1.3 删除索引

```
DELETE http://192.168.223.129:9200/blog1
```



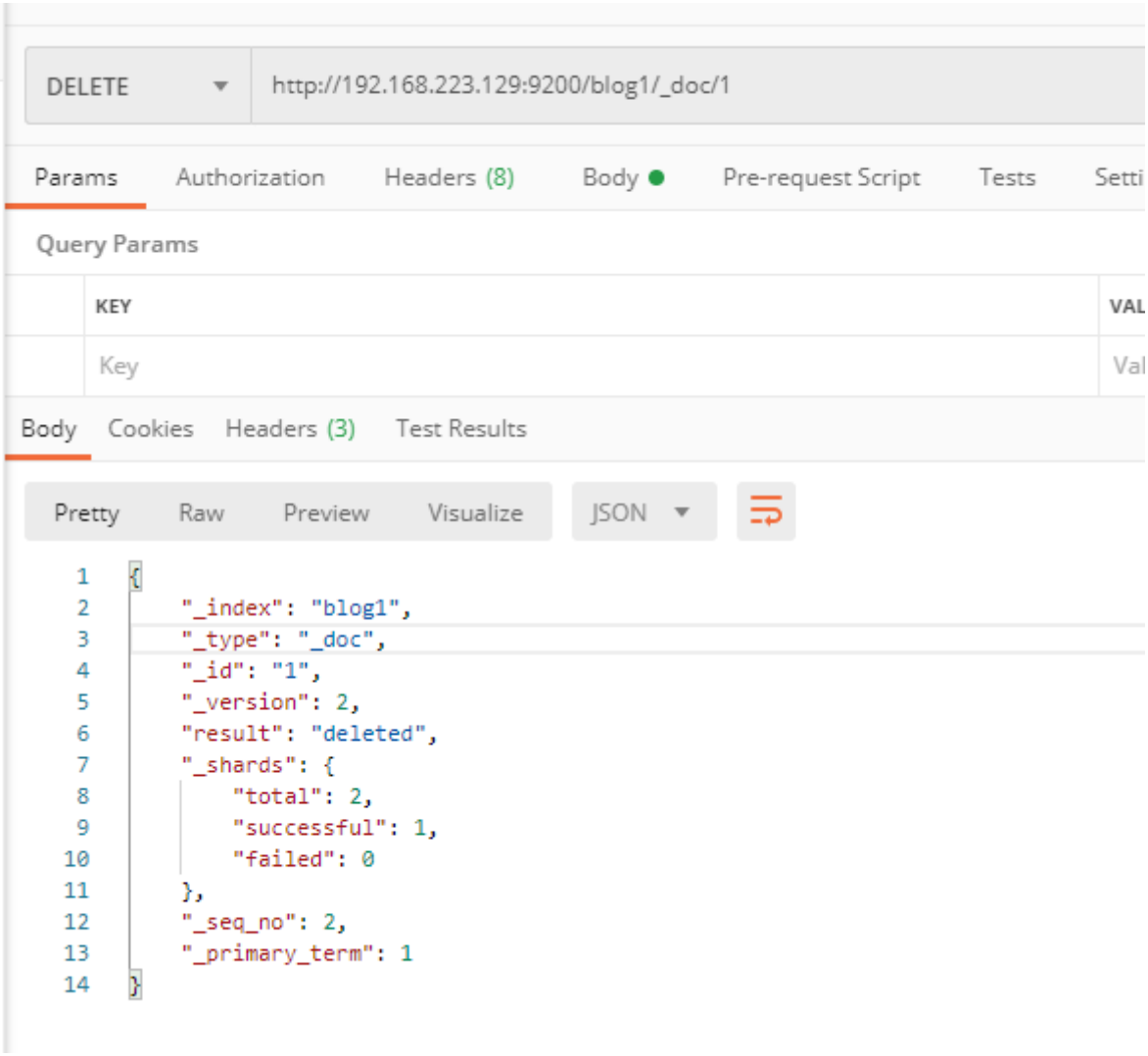
4.1.4 创建/修改文档

```
#请求uri blog1-数据库 _doc（数据表） 1(数据行号)
PUT http://192.168.223.129:9200/blog1/_doc/1
#请求体（对应索引mapper）
{
    "id": 1,
    "title": "ElasticSearch是一个基于Lucene的搜索服务器",
    "content": "它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。
Elasticsearch是用Java 开发的，并作为Apache许可条款下的开放源码发布，是当前流行的企业级搜索引
擎。设计用于云计算中，能够达到实时 搜索，稳定，可靠，快速，安装使用方便。"
}
```



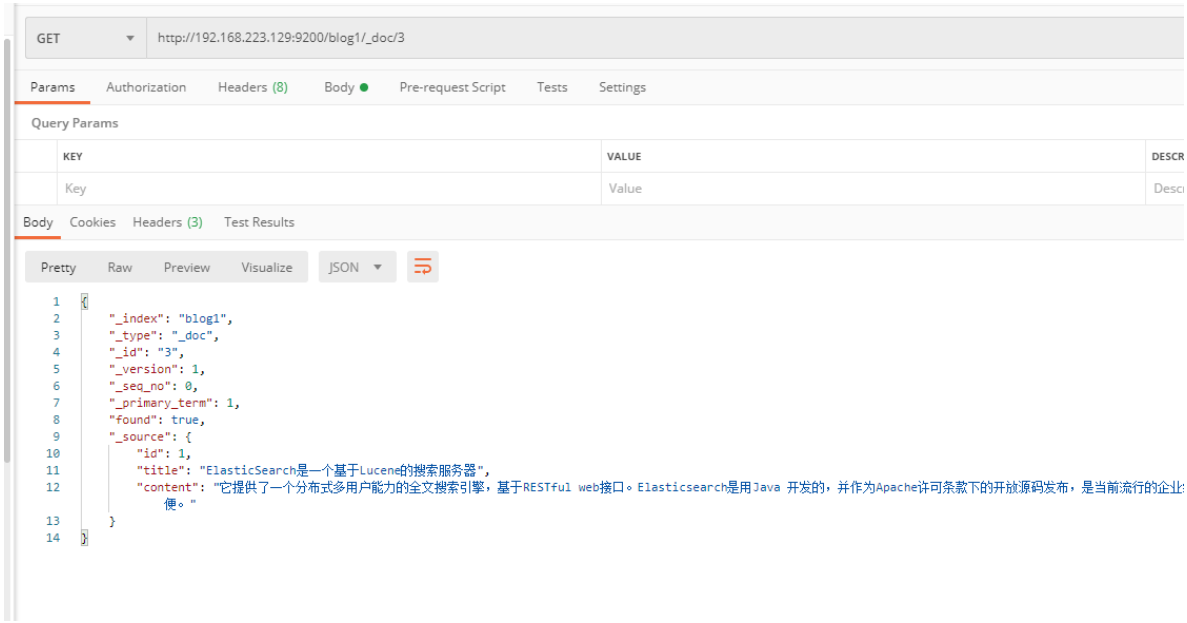
4.1.5 删除文档

```
DELETE http://192.168.223.129:9200/blog1/_doc/1
```



4.1.6 查询文档（ID）

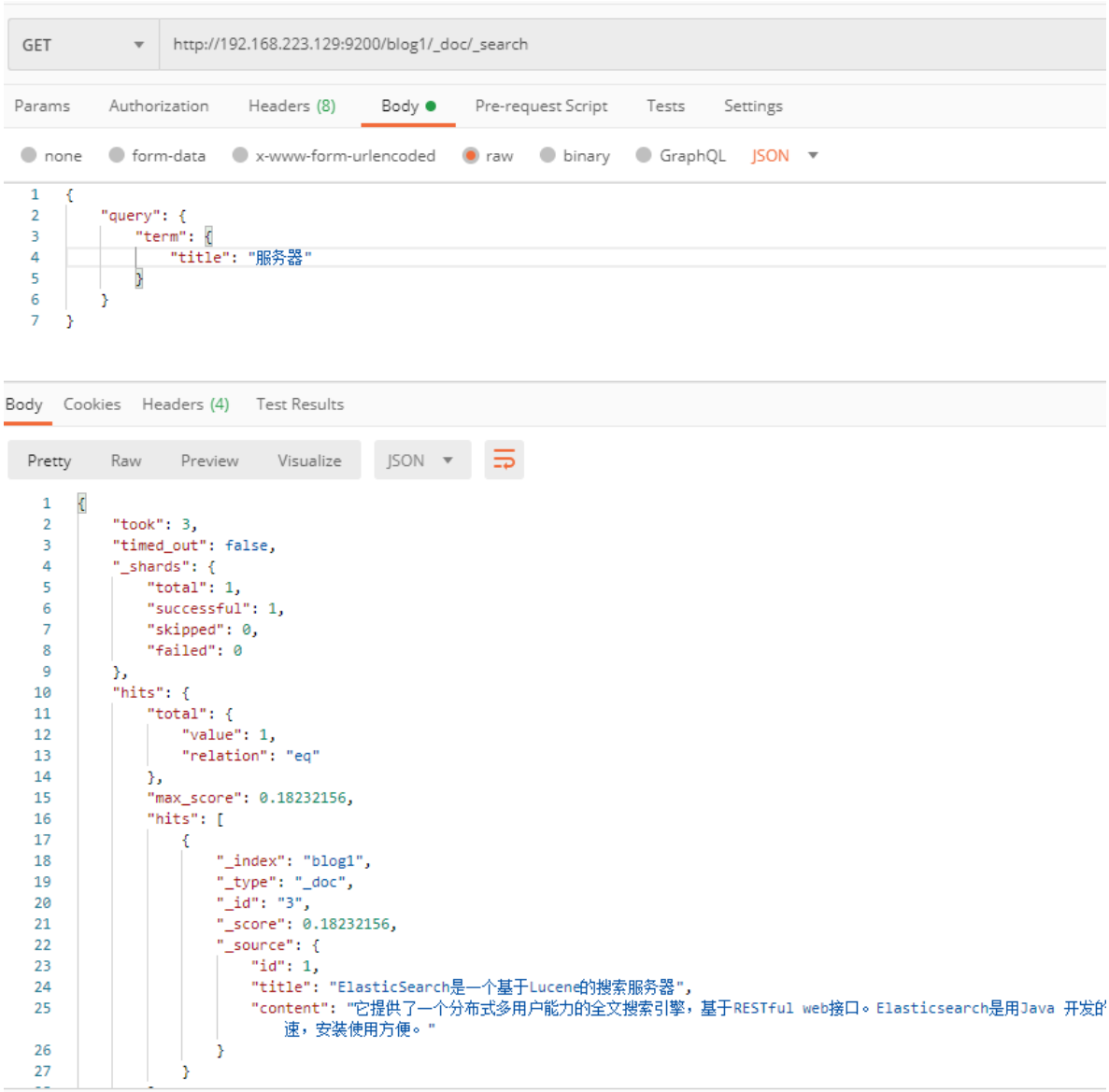
```
GET http://192.168.223.129:9200/blog1/_doc/1
```

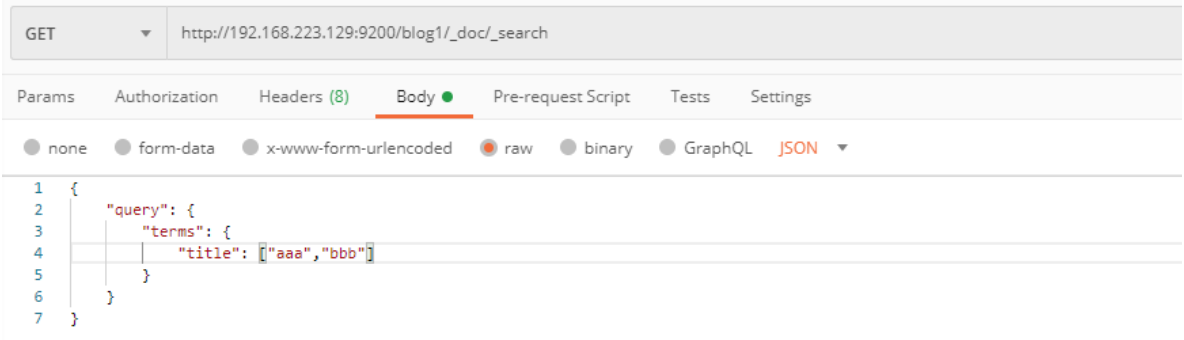


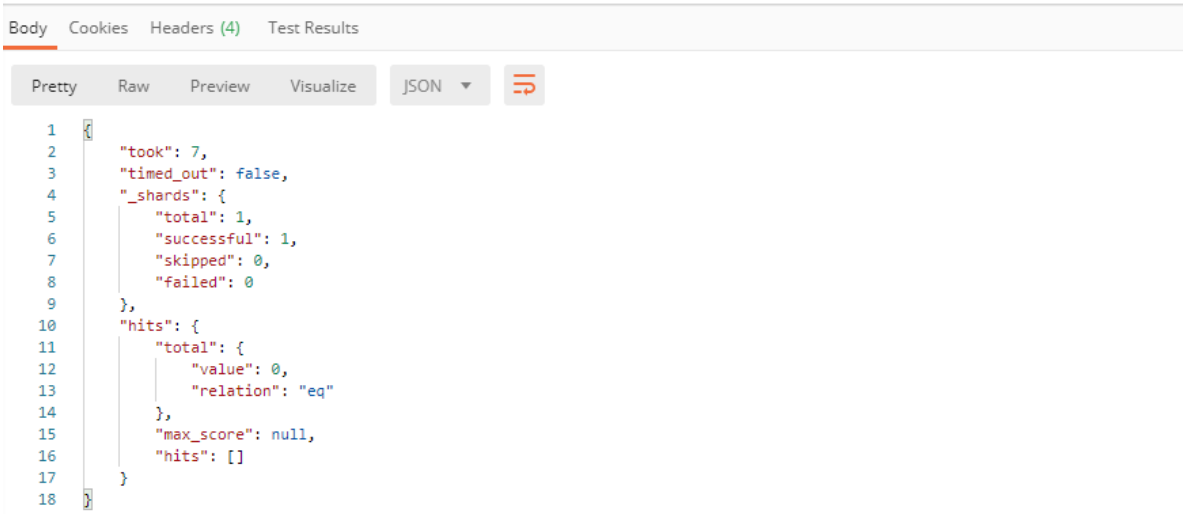
4.1.7 查询文档（词条）

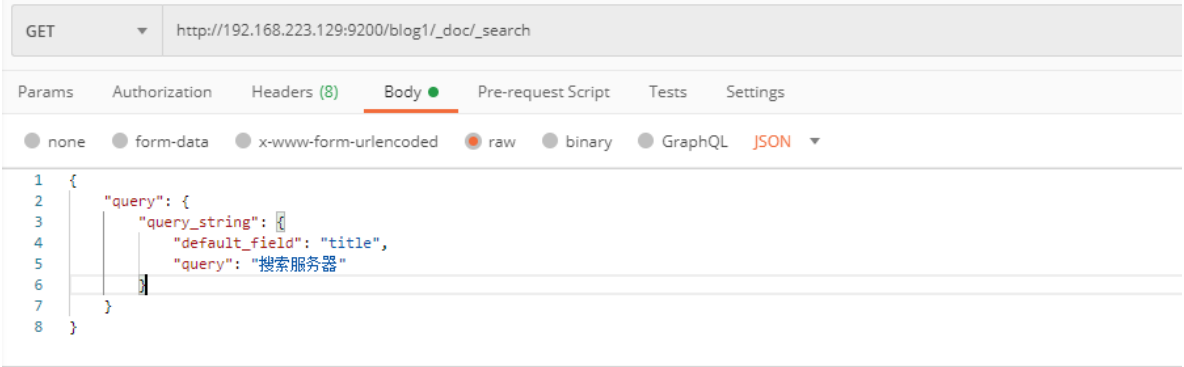
```
#请求URI
```

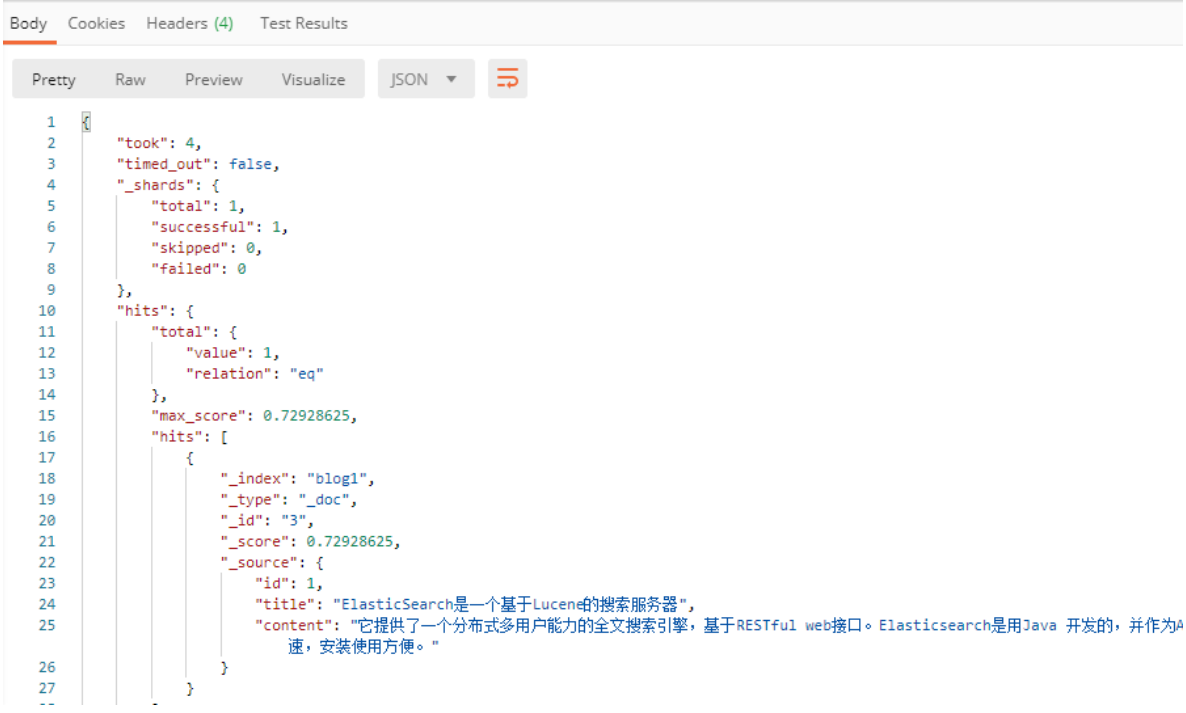
```
GET http://192.168.223.129:9200/blog1/_doc/_search
#请求体（单条件）
{
    "query": {
        "term": {
            "title": "服务器"
        }
    }
}

#请求体（单条件）
{
    "query": {
        "terms": {
            "title": ["服务器","aaa"]
        }
    }
}
#请求体（query_string）
{
    "query": {
        "query_string": {
            "default_field": "title",
            "query": "搜索服务器"
        }
    }
}
```


GET
http://192.168.223.129:9200/blog1/_doc/_search
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"query": {
"term": {
"title": "服务器"
}
}
}
Body
Cookies
Headers (4)
Test Results
Pretty
Raw
Preview
Visualize
JSON
{
"took": 3,
"timed_out": false,
"_shards": {
"total": 1,
"successful": 1,
"skipped": 0,
"failed": 0
},
"hits": {
"total": {
"value": 1,
"relation": "eq"
},
"max_score": 0.18232156,
"hits": [
{
"_index": "blog1",
"_type": "_doc",
"_id": "3",
"_score": 0.18232156,
"_source": {
"id": 1,
"title": "ElasticSearch是一个基于Lucene的搜索服务器",
"content": "它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。Elasticsearch是用Java 开发的
速，安装使用方便。"
}
}









4.2 IK分词器与标准分词器的区别

上面创建的索引映射为IK分词器

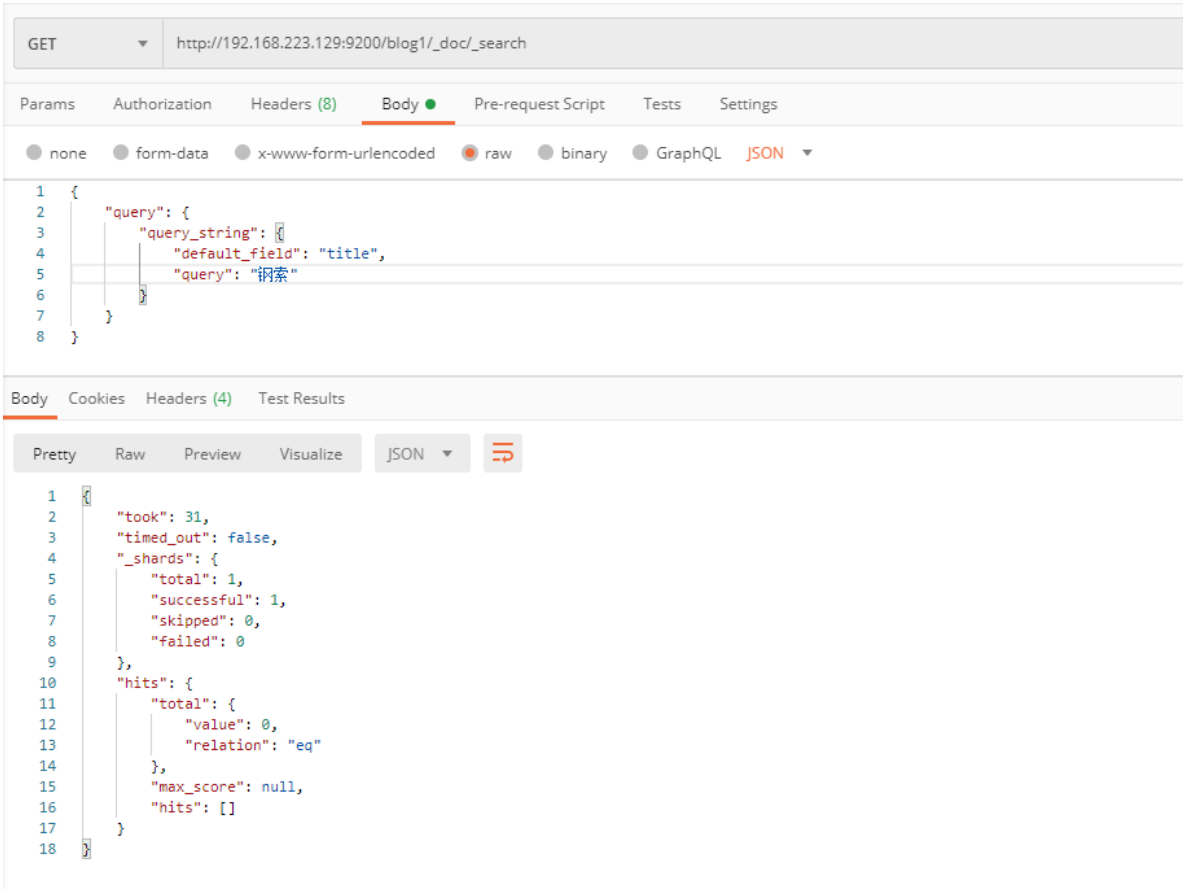
现在我们使用标准分词器再次创建一个blog2索引index

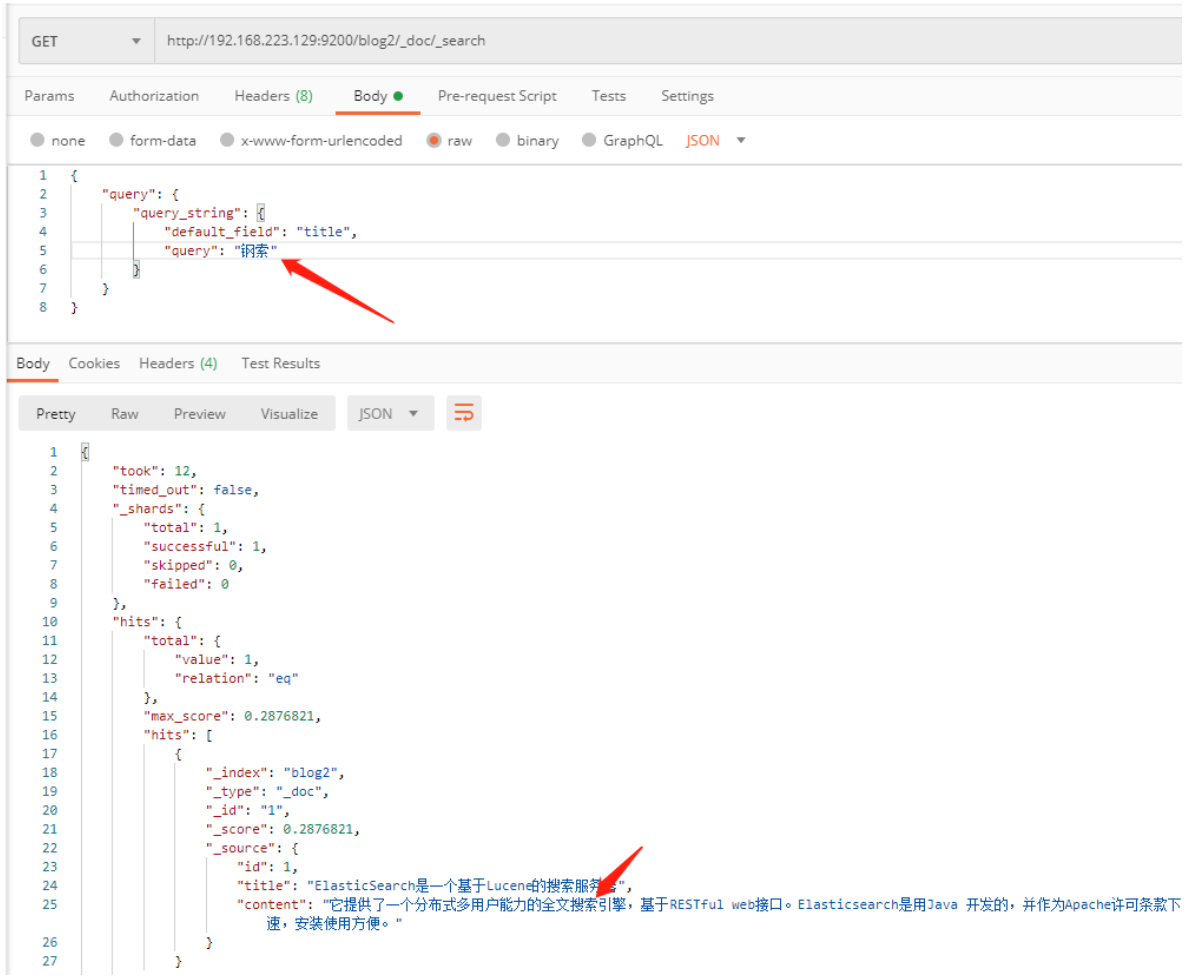
```
请求url：
PUT http://192.168.223.129:9200/blog2
请求体：
{
    "mappings": {
        "properties": {
            "id": {
                "type": "long"
            },
            "title": {
                "type": "text",
                "analyzer": "standard"
            },
            "content": {
                "type": "text",
                "analyzer": "standard"
            }
        }
    }
}
```

然后往里面创建文档对象，保证内容一模一样

```
#请求uri blog1-数据库 _doc（数据表） 1(数据行号)
PUT http://192.168.223.129:9200/blog2/_doc/1
#请求体（对应索引mapper）
{
    "id": 1,
    "title": "ElasticSearch是一个基于Lucene的搜索服务器",
    "content": "它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。
Elasticsearch是用Java 开发的，并作为Apache许可条款下的开放源码发布，是当前流行的企业级搜索引
擎。设计用于云计算中，能够达到实时 搜索，稳定，可靠，快速，安装使用方便。"
}
```

现在测试





可以发现标准分词器搜索“钢索”的时候，能够搜出结果，而IK分词器不可以！

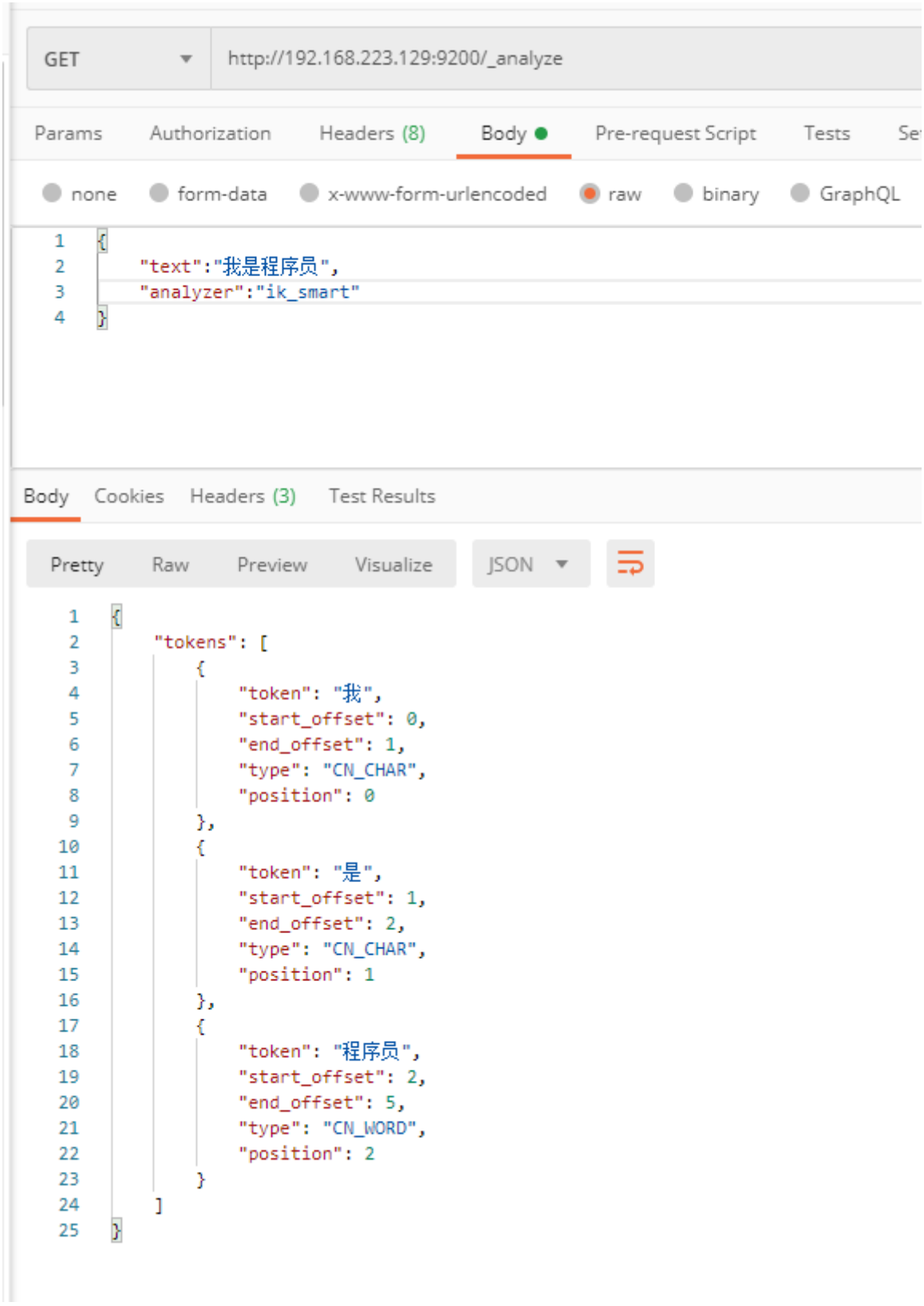
原因在于：

标准分词器会将每一个字符拆分成查询条件，然后取并集

IK分词器会根据字典里面是否有该词语来判断是否作为条件！

IK分词器分词方式又包括两种：

ik_smart：最小切分，比如"我是程序员"这个词语会被切分为"我"，"是"，“程序员”



ik_max_word：最细切分，比如"我是程序员"这个词语会被切分为"我"，"是"，“程序员”，“程序”，“员”

GET http://192.168.223.129:9200/_analyze

```
{
    "text":"我是程序员",
    "analyzer":"ik_max_word"
}
```

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "tokens": [
        {
            "token": "我",
            "start_offset": 0,
            "end_offset": 1,
            "type": "CN_CHAR",
            "position": 0
        },
        {
            "token": "是",
            "start_offset": 1,
            "end_offset": 2,
            "type": "CN_CHAR",
            "position": 1
        },
        {
            "token": "程序员",
            "start_offset": 2,
            "end_offset": 5,
            "type": "CN_WORD",
            "position": 2
        },
        {
            "token": "程序",
            "start_offset": 2,
            "end_offset": 4,
            "type": "CN_WORD",
            "position": 3
        },
        {
            "token": "员",
            "start_offset": 4,
            "end_offset": 5,
            "type": "CN_CHAR",
            "position": 4
```

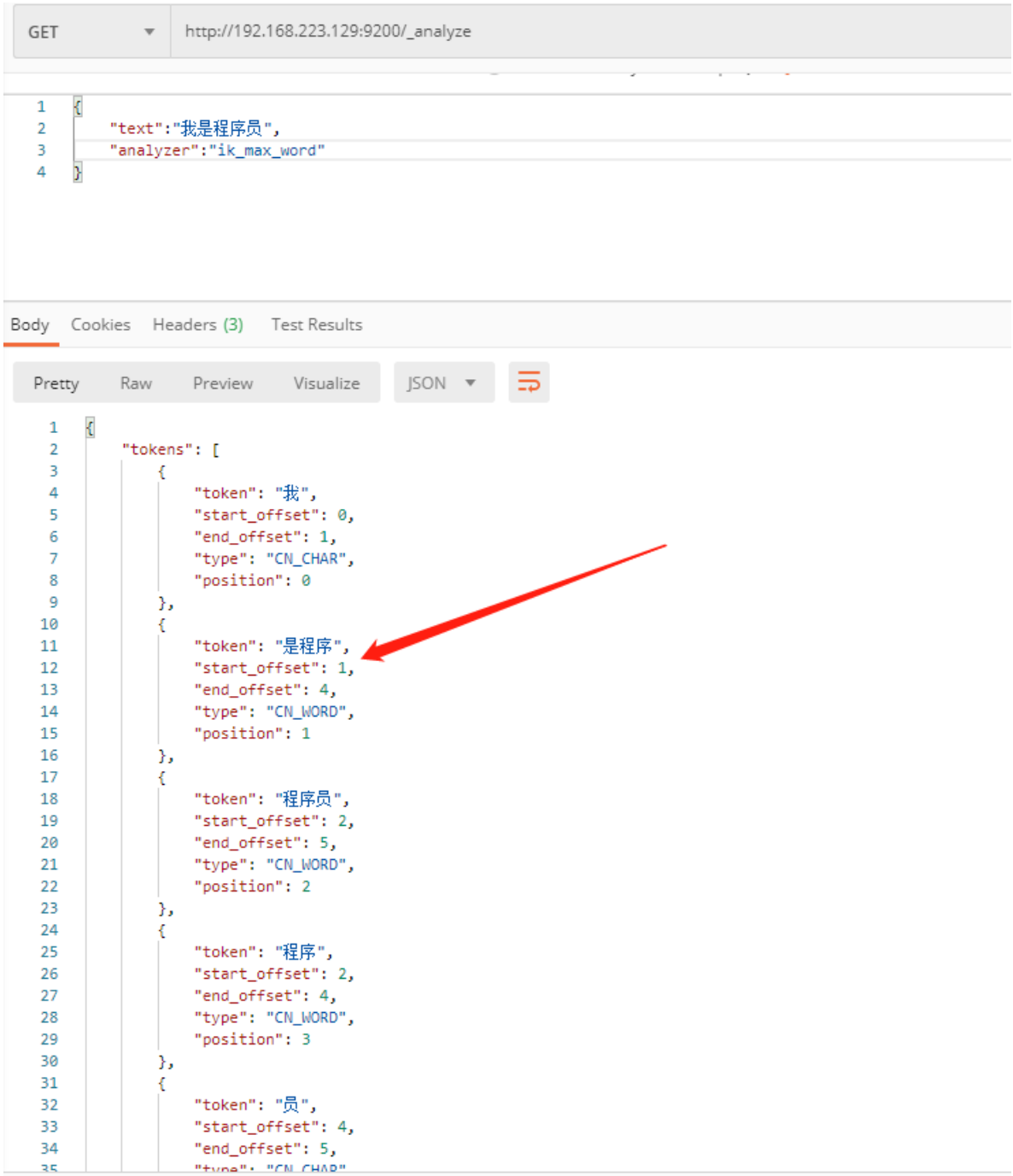
我们修改一下IK分词器的字典：

```
vim /usr/share/elasticsearch/plugins/analysis-ik/config/IKAnalyzer.cfg.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
        <comment>IK Analyzer 扩展配置</comment>
        <!--用户可以在这里配置自己的扩展字典 -->
        <entry key="ext_dict">laohu-dict.dic</entry>
         <!--用户可以在这里配置自己的扩展停止词字典-->
        <entry key="ext_stopwords"></entry>
        <!--用户可以在这里配置远程扩展字典 -->
        <!-- <entry key="remote_ext_dict">words_location</entry> -->
        <!--用户可以在这里配置远程扩展停止词字典-->
        <!-- <entry key="remote_ext_stopwords">words_location</entry> -->
</properties>
~
```

```
vim laohu-dict.dic

#增加如下词汇
是程序
```

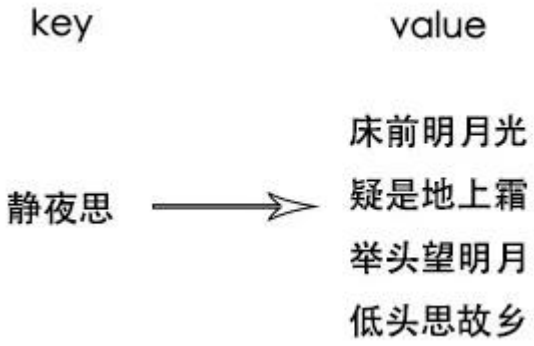
再次按最细拆分查询(最小拆分会认为“是程序”这个词并不是一个有用的搜索条件):



PS：两种分词器使用的最佳场景是：索引时用ik_max_word，在搜索时用ik_smart。
即：索引时最大化的将文章内容分词，搜索时更精确的搜索到想要的结果。

5、ElasticSearch原理

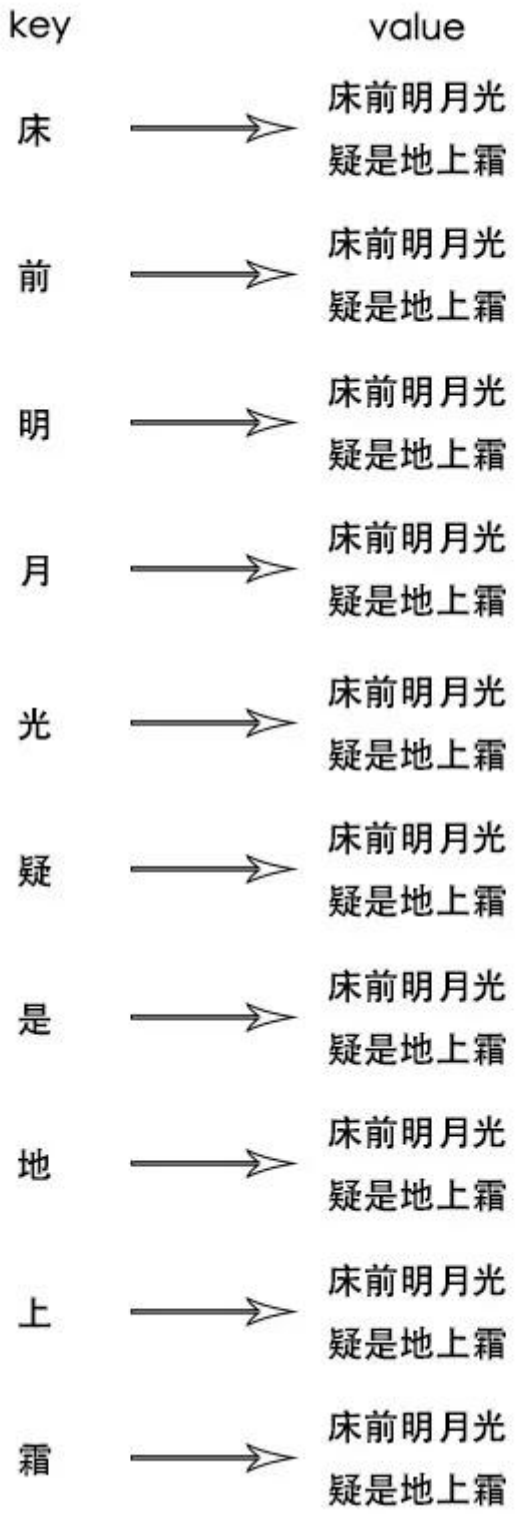
ES使用的倒排索引，既然有倒排索引，那就有正向索引，其实我们使用的Map也可以理解成一种正向索引，比如下面这首诗：



这个就是正向索引，提问者会让你背诵静夜思，你会很流利的背诵出来，因为你的脑海里有这么一个K/V的索引概念，但是提问者让你背诵包含"前"字的诗句

你的第一反应肯定是懵逼，有这首诗吗？真的有吗？想不起来，为什么，因为你的脑海由于没有该索引，你只能遍历脑海中所有诗词，当你的脑海中诗词量大的时候，就很难在短时间内得到结果了

如果你在你的脑海里面建立这么一组索引：



这样，你可以索引到对应的诗句，但是这样的话，会造成大量的索引数据，你的脑袋会爆炸，如果把这个索引改一下：

这样，一个诗题带表整个诗体，是不是就压缩了很大的空间呢？而"静夜思"在你的脑袋里面本来就已经存在正向索引了，这样，整个环节就串起来了！

总结一下搜索引擎三个过程：**爬取内容，进行分词，建立倒排索引**