

Elasticsearch第一天

学习目标:

1. 理解Elasticsearch的作用
2. 能够安装Elasticsearch服务
3. 理解Elasticsearch的相关概念
4. 能够使用Postman发送Restful请求操作Elasticsearch
5. 理解分词器的作用
6. 能够使用Elasticsearch集成IK分词器
7. 能够完成es集群搭建

第一章 Elasticsearch简介

1.1 什么是Elasticsearch

Elasticsearch，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使用Java开发。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。

1.2 Elasticsearch的使用案例

- 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 构建挖财自己的日志采集和分析体系
- 2013年初，GitHub抛弃了Solr，采用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”

第二章 Elasticsearch安装与启动

2.1 下载ES压缩包

Elasticsearch分为Linux和Window版本，基于我们主要学习的是Elasticsearch的Java客户端的使用，所以我们课程中使用的是安装较为简便的Window版本，项目上线后，公司的运维人员会安装Linux版的ES供我们连接使用。

Elasticsearch的官方地址：<https://www.elastic.co/cn/downloads/past-releases>


← → ↻ https://www.elastic.co/cn/products/elasticsearch

elastic 产品 云端 服务 客户 了解

下载 联系我们 Q CN

Elasticsearch


Download As A Service



ELASTICSEARCH
The Elastic Stack

Elastic Stack 的核心

Elasticsearch 是一个分布式、RESTful 风格的搜索和分析引擎，能够解决不断涌现出的各种用例。作为 Elastic Stack 的核心，它集中存储您的数据，帮助您发现意料之中以及意料之外的情况。



← → × https://www.elastic.co/cn/downloads/elasticsearch

elastic 产品 云端 服务 客户 了解

下载 联系我们 Q CN

Downloads

Download Elasticsearch

Want to upgrade? We'll give you a hand. [Upgrade Guidance](#)

Version: 7.0.1

Release date: May 02, 2019

License: [Elastic License](#)

Downloads: [Windows sha512](#) [MacOS sha512](#)
[Linux sha512](#) [Debian sha512](#)
[RPM sha512](#) [MSI \(BETA\) sha512](#)

Containers: Run with [Docker](#)

Notes: Running on Kubernetes? Try the Elasticsearch [Helm chart \(alpha\)](#).
This default distribution is governed by the Elastic License, and includes the [full set of free features](#).

View the detailed release notes [here](#).
Not the version you're looking for? [View past releases](#).
The pure Apache 2.0 licensed distribution is available [here](#).
You can obtain the source code for the bundled JDK [here](#).
The unbounded distribution is available [here](#).

正在等待 js.clearbit.com 的响应...

↻ https://www.elastic.co/cn/downloads/past-releases

elastic 产品 云端 服务 客户 了解

下载 联系我们 Q CN

Downloads

Past Releases

✕ Clear All Filters Elasticsearch Versions

Elasticsearch 6.7.2 May 03, 2019	See Release Notes Download
Elasticsearch 7.0.1 May 02, 2019	See Release Notes Download
Elasticsearch 7.0.0 April 11, 2019	See Release Notes Download
Elasticsearch 6.7.1 April 05, 2019	See Release Notes Download

Downloads

Elasticsearch 5.6.8

[ZIP sha](#)[TAR sha](#)[DEB sha](#)[RPM sha](#)[MSI \(BETA\) sha](#)[See issues on GitHub](#)

在资料中已经提供了下载好的5.6.8的压缩包：



elasticsearch-5.
6.8.zip

2.2 安装ES服务

Window版的Elasticsearch的安装很简单，类似Window版的Tomcat，解压开即安装完毕，解压后的Elasticsearch的目录结构如下：

bin	可执行二进制文件	2018/3/13 10:08	文件夹
config	配置信息目录	2018/3/13 10:08	文件夹
lib	jar包存放目录	2018/3/13 10:08	文件夹
logs	日志存在目录	2018/3/13 10:08	文件夹
modules	模块存在目录	2018/3/13 10:08	文件夹
plugins	插件安装目录	2018/3/13 10:08	文件夹
LICENSE.txt		2018/3/13 10:02	文本文档 12 KB
NOTICE.txt		2018/3/13 10:07	文本文档 188 KB
README.textile		2018/3/13 10:02	TEXTILE 文件 10 KB

2.3 启动ES服务

点击Elasticsearch下的bin目录下的Elasticsearch.bat启动，控制台显示的日志信息如下：

elasticsearch	2018/3/13 16:15	文件	8 KB
elasticsearch.bat	2018/2/16 16:43	Windows 批处理...	4 KB
elasticsearch.in.bat	2018/3/13 16:17	Windows 批处理...	1 KB
elasticsearch.in.sh	2018/2/16 16:43	SH 文件	1 KB
elasticsearch-keystore	2018/2/16 16:43	文件	3 KB
elasticsearch-keystore.bat	2018/2/16 16:43	Windows 批处理...	1 KB
elasticsearch-plugin	2018/2/16 16:43	文件	3 KB
elasticsearch-plugin.bat	2018/2/16 16:43	Windows 批处理...	1 KB
elasticsearch-service.bat	2018/2/16 16:43	Windows 批处理...	11 KB
elasticsearch-service-mgr.exe	2018/2/16 16:43	应用程序	102 KB
elasticsearch-service-x64.exe	2018/2/16 16:43	应用程序	102 KB
elasticsearch-service-x86.exe	2018/2/16 16:43	应用程序	79 KB
elasticsearch-systemd-pre-exec	2018/2/16 16:43	文件	1 KB
elasticsearch-translog	2018/2/16 16:43	文件	3 KB
elasticsearch-translog.bat	2018/2/16 16:43	Windows 批处理...	2 KB

```

Elasticsearch 5.6.8
[2018-04-04T14:02:40,913][INFO ][o.e.n.Node ] JVM arguments [-Xms2g, -Xmx2g, -XX:+UseConcMarkSweepGC, -XX:CMSInitiatingOccupancyFraction=75, -XX:+UseCMSInitiatingOccupancyOnly, -XX:+AlwaysPreTouch, -Xss1m, -Djava.awt.headless=true, -Dfile.encoding=UTF-8, -Djna.nosys=true, -Djdk.io.permissionsUseCanonicalPath=true, -Dio.netty.noUnsafe=true, -Dio.netty.noKeySetOptimization=true, -Dio.netty.recycler.maxCapacityPerThread=0, -Dlog4j.shutdownHookEnabled=false, -Dlog4j2.disable.jmx=true, -Dlog4j.skipJansi=true, -XX:+HeapDumpOnOutOfMemoryError, -Delasticsearch, -Des.path.home=E:\elasticsearch-5.6.8]
[2018-04-04T14:02:41,545][INFO ][o.e.p.PluginsService ] [7TD9pkg] loaded module [aggs-matrix-stats]
[2018-04-04T14:02:41,545][INFO ][o.e.p.PluginsService ] [7TD9pkg] loaded module [ingest-common]
[2018-04-04T14:02:41,546][INFO ][o.e.p.PluginsService ] [7TD9pkg] loaded module [lang-expression]
[2018-04-04T14:02:41,546][INFO ][o.e.p.PluginsService ] [7TD9pkg] loaded module [lang-groovy]
[2018-04-04T14:02:41,546][INFO ][o.e.p.PluginsService ] [7TD9pkg] loaded module [lang-mustache]
[2018-04-04T14:02:41,546][INFO ][o.e.p.PluginsService ] [7TD9pkg] loaded module [lang-painless]
[2018-04-04T14:02:41,547][INFO ][o.e.p.PluginsService ] [7TD9pkg] loaded module [parent-join]
[2018-04-04T14:02:41,546][INFO ][o.e.p.PluginsService ] [7TD9pkg] loaded module [percolator]
[2018-04-04T14:02:41,546][INFO ][o.e.p.PluginsService ] [7TD9pkg] loaded module [reindex]
[2018-04-04T14:02:41,546][INFO ][o.e.p.PluginsService ] [7TD9pkg] loaded module [transport-netty3]
[2018-04-04T14:02:41,546][INFO ][o.e.p.PluginsService ] [7TD9pkg] loaded module [transport-netty4]
[2018-04-04T14:02:41,547][INFO ][o.e.p.PluginsService ] [7TD9pkg] no plugins loaded
[2018-04-04T14:02:42,881][INFO ][o.e.d.DiscoveryModule ] [7TD9pkg] using discovery type [zen]
[2018-04-04T14:02:43,261][INFO ][o.e.n.Node ] initialized
[2018-04-04T14:02:43,262][INFO ][o.e.n.Node ] starting ...
[2018-04-04T14:02:43,852][INFO ][o.e.t.TransportService ] [7TD9pkg] publish_address {127.0.0.1:9300}, bound_addresses {127.0.0.1:9300}
[2018-04-04T14:02:46,996][INFO ][o.e.c.s.ClusterService ] [7TD9pkg] new_master {7TD9pkg} {7TD9pkg:SfGYc6dB5JG1TA} {7pCQK-EszTXq38wDCrojd4fg} {127.0.0.1} (127.0.0.1:9300), reason: zen-disco-elected-as-master ([0] nodes joined)
[2018-04-04T14:02:47,100][INFO ][o.e.g.GatewayService ] [7TD9pkg] recovered [0] indices into cluster state
[2018-04-04T14:02:47,305][INFO ][o.e.h.n.Netty4HttpServerTransport] [7TD9pkg] publish_address {127.0.0.1:9200}, bound_addresses {127.0.0.1:9200}
[2018-04-04T14:02:47,305][INFO ][o.e.n.Node ] [7TD9pkg] started

```

注意：9300是tcp通讯端口，集群间和TCP 客户端都执行该端口，9200是http协议的RESTful接口。

通过浏览器访问Elasticsearch服务器，看到如下返回的json信息，代表服务启动成功：

```

127.0.0.1:9200
{
  "name": "node-1",
  "cluster_name": "my-elasticsearch",
  "cluster_uuid": "BFB1Q6syRrCH6is9L9Qqcg",
  "version": {
    "number": "5.6.8",
    "build_hash": "688ecce",
    "build_date": "2018-02-16T16:46:30.010Z",
    "build_snapshot": false,
    "lucene_version": "6.6.1"
  },
  "tagline": "You Know, for Search"
}

```

注意：Elasticsearch是使用java开发的，且本版本的es需要的jdk版本要是1.8及以上，所以安装Elasticsearch之前保证JDK1.8+安装完毕，并正确的配置好JDK环境变量，否则启动Elasticsearch失败。

2.4 安装ES的图形化界面插件

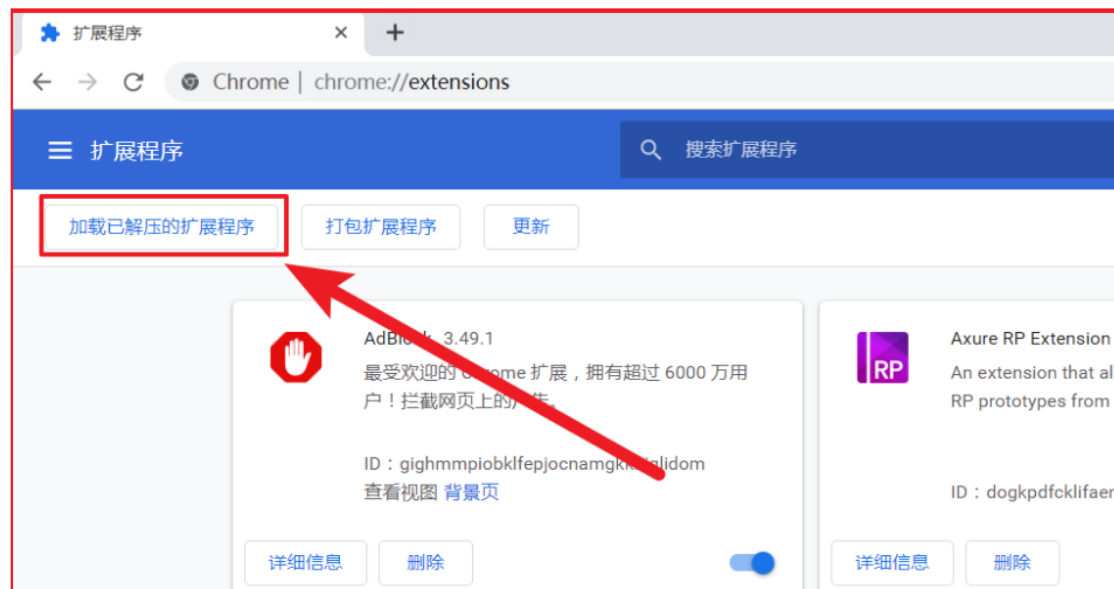
1. 在Chrome浏览器地址栏中输入: <chrome://extensions/>



2. 打开Chrome扩展程序的开发者模式
3. 将资料中的ElasticSearch-head-Chrome插件.crx拖入浏览器的插件页面



4. 解压crx插件, 通过加载已解压的扩展程序来加载



5. 最后即可安装成功

第三章 Elasticsearch核心概念

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:

MySQL (relational DB)	-> Databases (数据库)	-> Tables (表)	-> Rows (行)	-> Columns (列)
Elasticsearch	-> Indexes (索引)	-> Types (类型)	-> Documents (文档)	-> Fields (字段)

3.1 索引

索引 index, 在Elasticsearch中存储数据的行为就叫做索引(indexing), 一个索引由一个名字来标识(必须全部是小写字母的)。索引(index)这个词在Elasticsearch中有着不同的含义, 所以有必要区分:

- 索引（名词） 如上文所述，一个索引(index)就像是传统关系数据库中的数据库，它是相关文档存储的地方，index的复数是indices 或indexes。
- 索引（动词） 「索引一个文档」表示把一个文档存储到索引（名词）里，以便它可以被检索或者查询。这很像SQL中的 INSERT 关键字。它与Insert差别是，如果文档已经存在，新的文档将覆盖旧的文档。

3.2 类型

类型 type，在一个索引中，可以定义一种或多种类型。相当于关系型数据库中的表。一个类型是你索引的一个逻辑上的分类或分区，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(用户表)，为博客数据定义另一个类型(Article表)，当然，也可以为评论数据定义另一个类型(评论表)。

3.3 字段

字段Field，相当于数据表字段，对不同属性进行分类标识

3.4 映射

映射 mapping是处理数据的方式和规则，是针对字段做的一些限制，字段的配置。比如：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。

3.5 文档

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：文档必须被赋予一个索引的type。

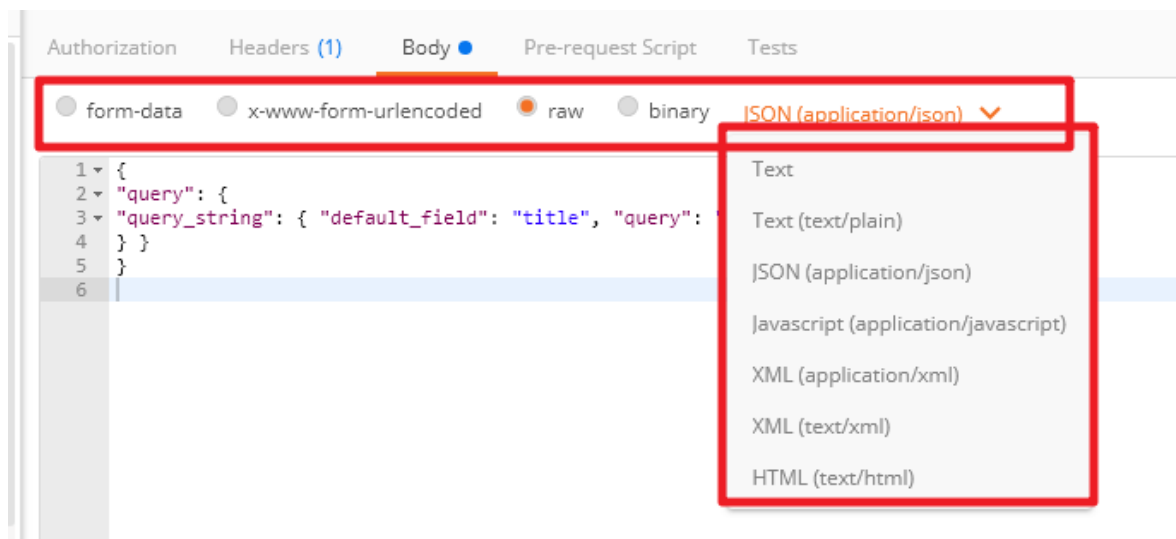
第四章 Elasticsearch的操作

实际开发中，主要有四种方式操作Elasticsearch服务：

- 第一种，Elasticsearch的RESTful接口直接操作。
- 第二种，Elasticsearch提供的Java API进行操作。
- 第三种，Spring Data Elasticsearch 进行操作。
- 第四种，ElasticSearch-head-master图形化界面操作。

4.1 安装Postman工具

Postman中文版是Postman这款强大网页调试工具的windows客户端，提供功能强大的Web API 和 HTTP 请求调试。软件功能强大，界面简洁明晰、操作方便快捷，设计得很人性化。Postman中文版能够发送任何类型的HTTP 请求 (GET, HEAD, POST, PUT..)，不仅能够表单提交，且可以附带任意类型请求体。



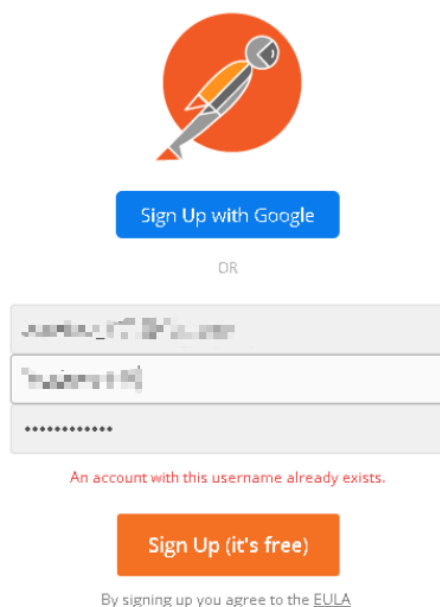
4.2 下载Postman工具

Postman官网: <https://www.getpostman.com>

课程资料中已经提供了安装包

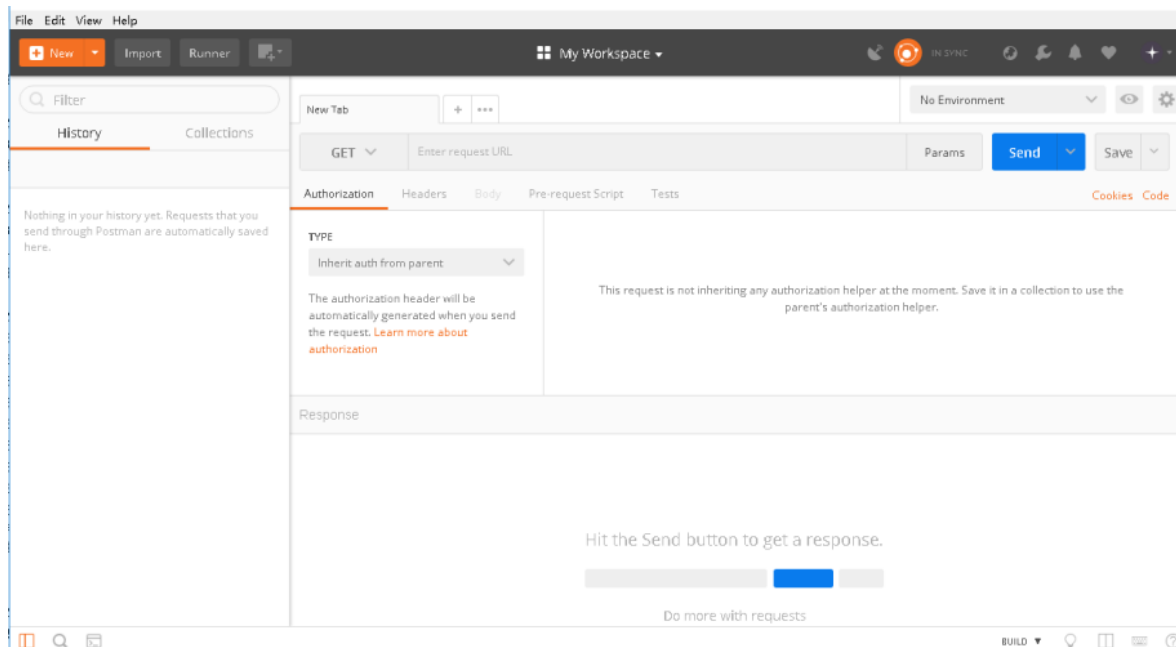


4.3 注册Postman工具



Why Sign Up

- ✓ Organize all your API development within Postman Workspaces
- ✓ Sync your Postman data across devices
- ✓ Backup your Postman data
- ✓ Create Documentation pages, Monitors and Mock Servers



4.4 使用Postman工具进行RESTful接口访问

4.4.1 Elasticsearch的RESTful接口

请求方法: <verb>

请求地址: <Protocol>://<Host>:<Port>/<Path>?<Query_String>

请求体: <Body>

其中:

参数	解释
VERB	适当的 HTTP 方法或 谓词: GET、POST、PUT、HEAD 或者 DELETE。
PROTOCOL	http 或者 https (如果你在 Elasticsearch 前面有一个 https 代理)
HOST	Elasticsearch 集群中任意节点的主机名, 或者用 localhost 代表本地机器上的节点。
PORT	运行 Elasticsearch HTTP 服务的端口号, 默认是 9200。
PATH	Path API 的终端路径 (例如 _count 将返回集群中文档数量)。Path 可能包含多个组件, 例如: _cluster/stats 和 _nodes/stats/jvm。
QUERY_STRING	任意可选的查询字符串参数 (例如 ?pretty 将格式化地输出 JSON 返回值, 使其更容易阅读)
BODY	一个 JSON 格式的请求体 (如果请求需要的话)

在学习数据库时,

创建数据库, 创建表(配置), 对表做增删改查, 简单查询, 分页查询, 复杂查询(Or, And, Between)

4.4.2 创建索引

请求方式: PUT
请求url: http://localhost:9200/blog1

postman截图:



Elasticsearch-head查看:



4.4.2 创建索引时配置映射

请求方式: PUT
请求url: http://localhost:9200/blog1

请求体:

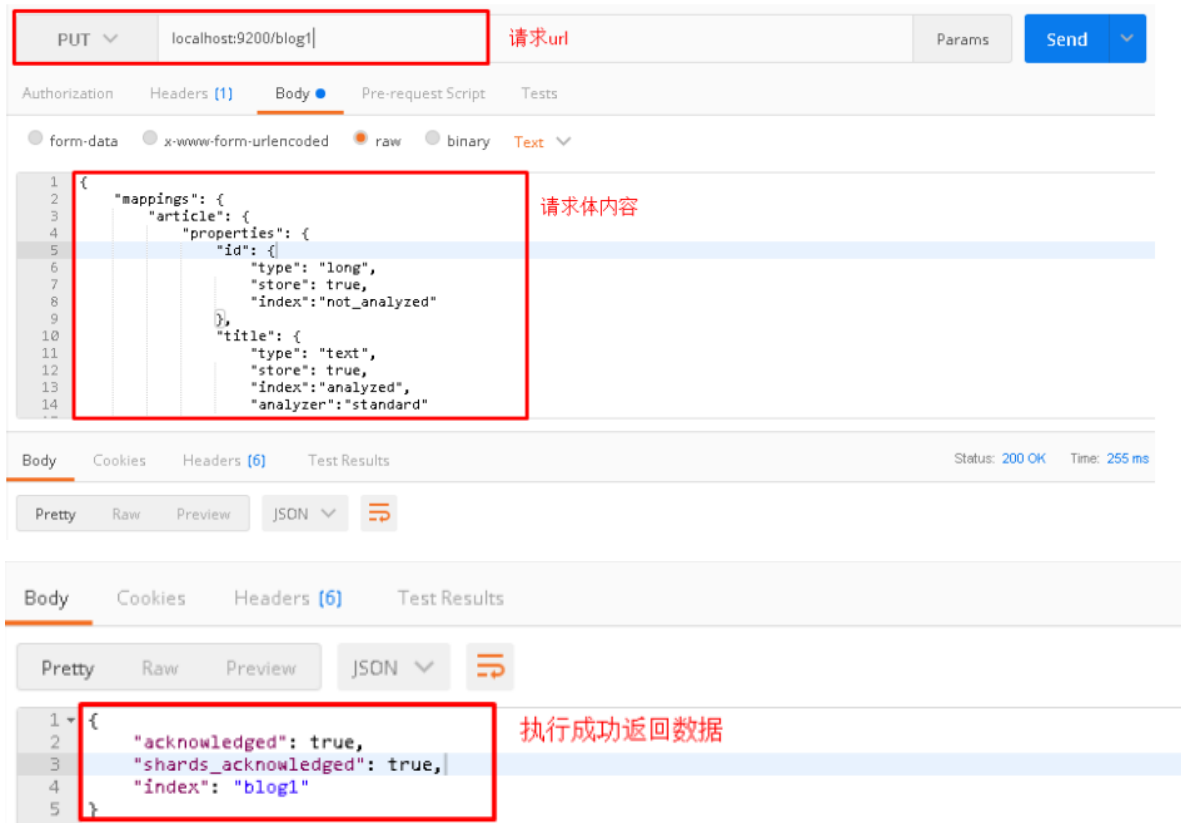
```
{
  "mappings": {
    "article": {
      "properties": {
        "id": {
          "type": "long",
          "store": true,
          "index": "not_analyzed"
        },
        "title": {
          "type": "text",
          "store": true,
          "index": "analyzed",
          "analyzer": "standard"
        },
        "content": {
          "type": "text",
```

```

    "store": true,
    "index": "analyzed",
    "analyzer": "standard"
  }
}
}
}
}

```

postman截图:



Elasticsearch-head查看:



4.4.3 创建索引后配置映射

我们可以在创建索引时设置mapping信息，当然也可以先创建索引然后再设置mapping。

在上一个步骤中不设置mapping信息，直接使用put方法创建一个索引，然后设置mapping信息。

路径中有带“_”的表示的是Path API。

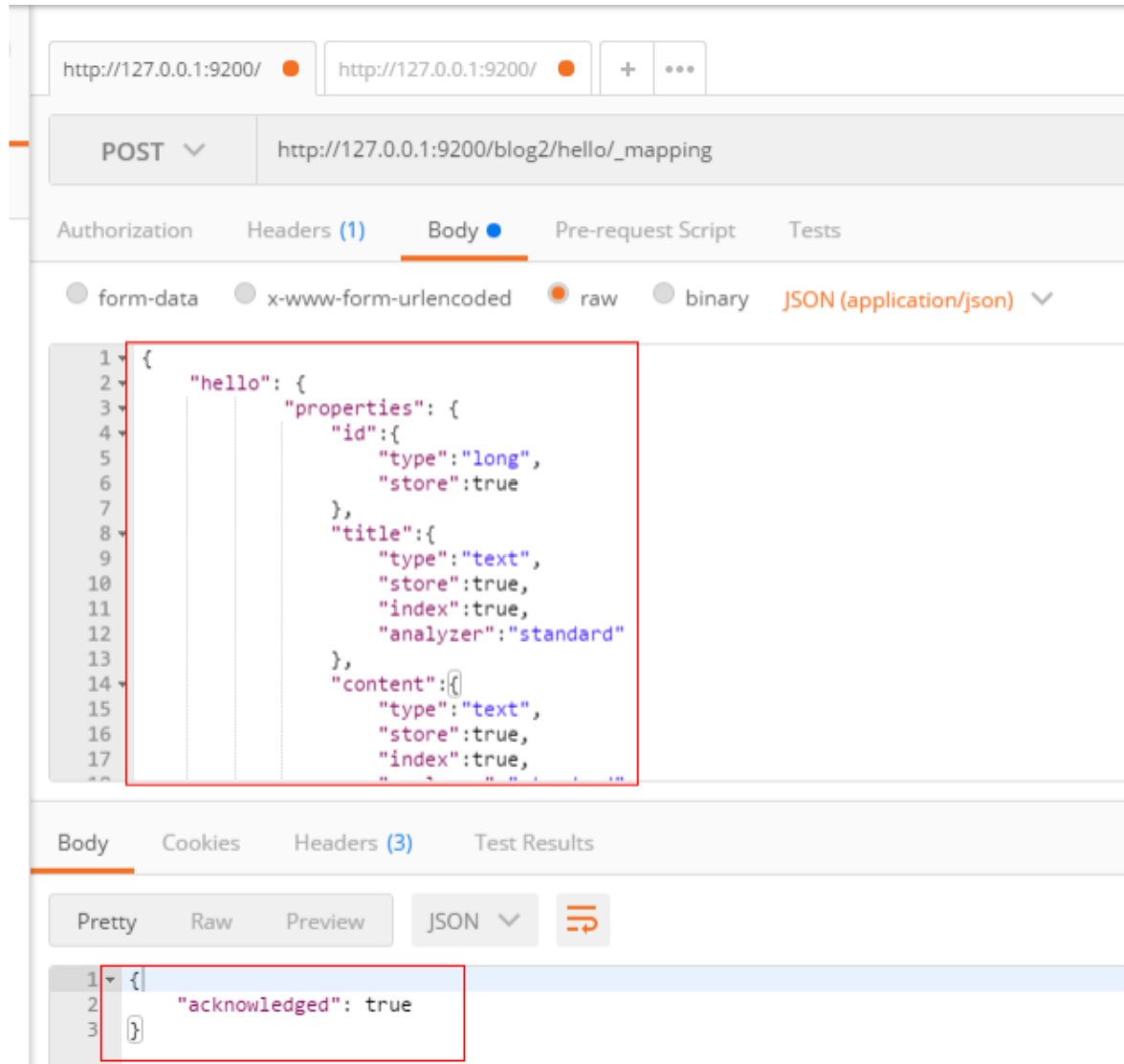
请求方式: POST

请求url: `http://127.0.0.1:9200/blog1/hello/_mapping`

请求体:

```
{
  "hello": {
    "properties": {
      "id": {
        "type": "long",
        "store": true
      },
      "title": {
        "type": "text",
        "store": true,
        "index": true,
        "analyzer": "standard"
      },
      "content": {
        "type": "text",
        "store": true,
        "index": true,
        "analyzer": "standard"
      }
    }
  }
}
```

PostMan截图

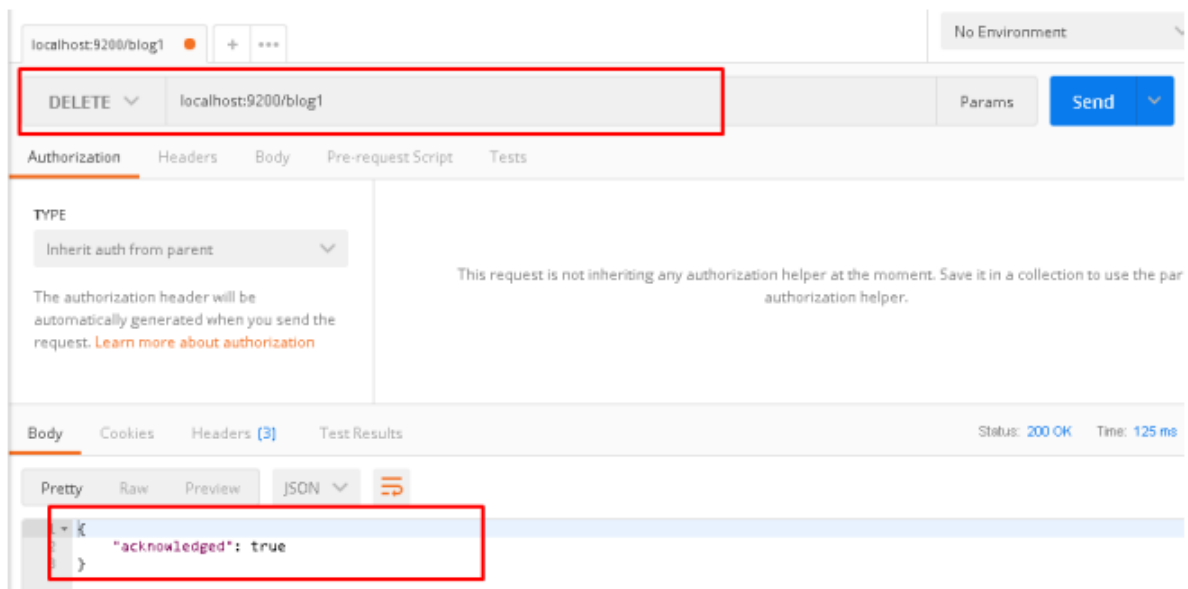


4.4.4 删除索引

请求方式: DELETE

请求url: `http://localhost:9200/blog1`

postman截图:



Elasticsearch-head查看:



4.4.5 创建文档

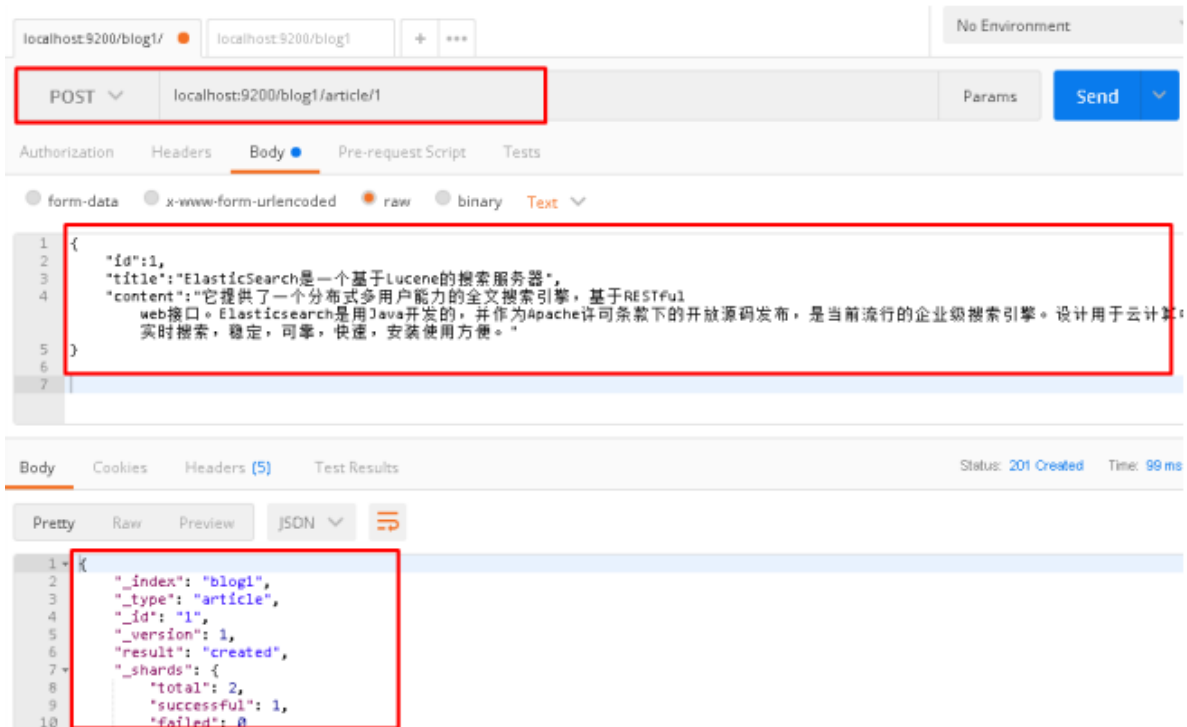
地址最后的值为索引库中文档的ID，如果不传会随机生成一个。

请求方式: POST
请求url: http://localhost:9200/blog1/article/1

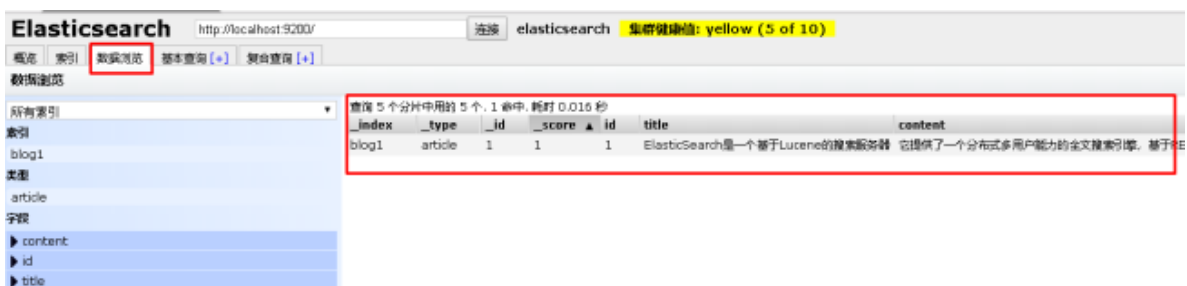
请求体:

```
{
  "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。"
}
```

postman截图:



Elasticsearch-head查看:



4.4.6 修改文档

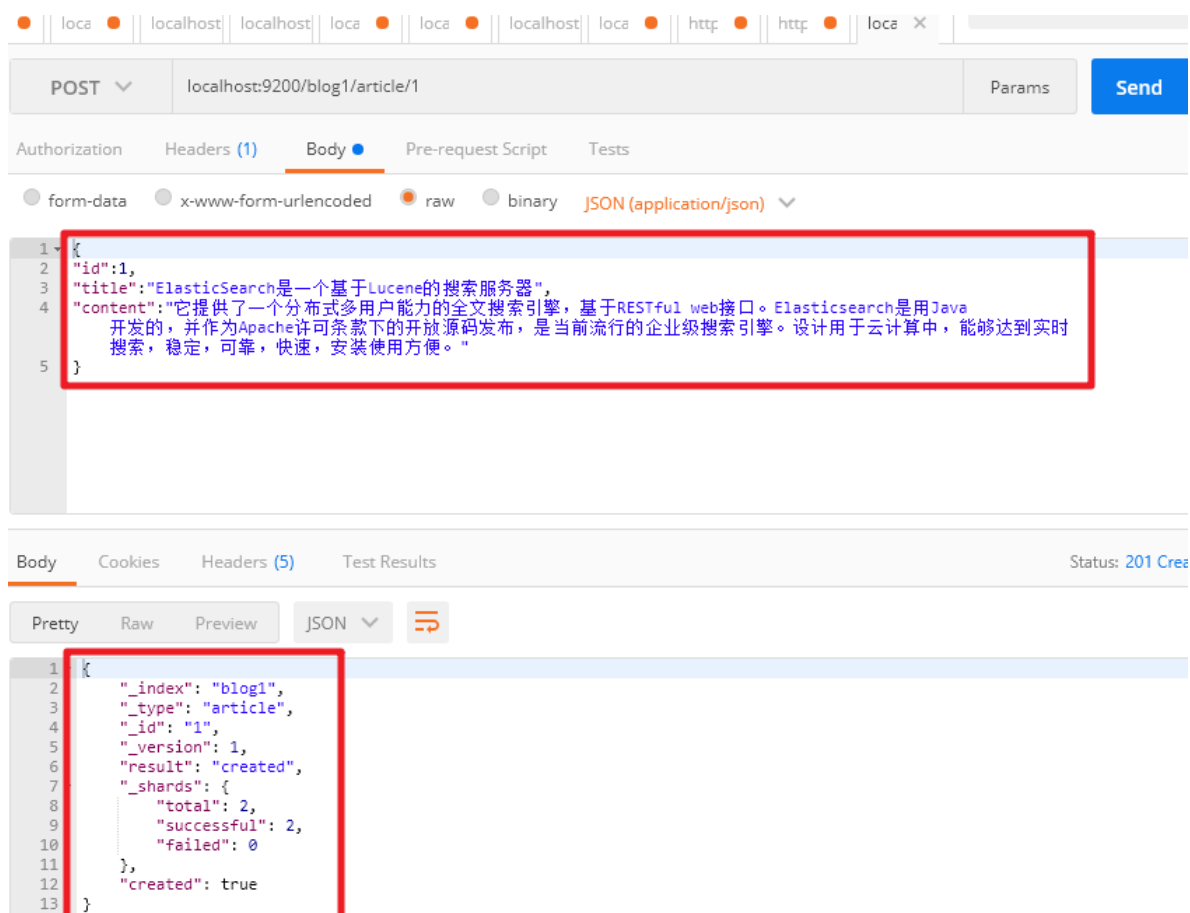
请求方式: POST

请求url: `http://localhost:9200/blog1/article/1`

请求体:

```
{
  "id": 1,
  "title": "【修改】Elasticsearch是一个基于Lucene的搜索服务器",
  "content": "【修改】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。"
}
```

postman截图:



Elasticsearch-head查看:

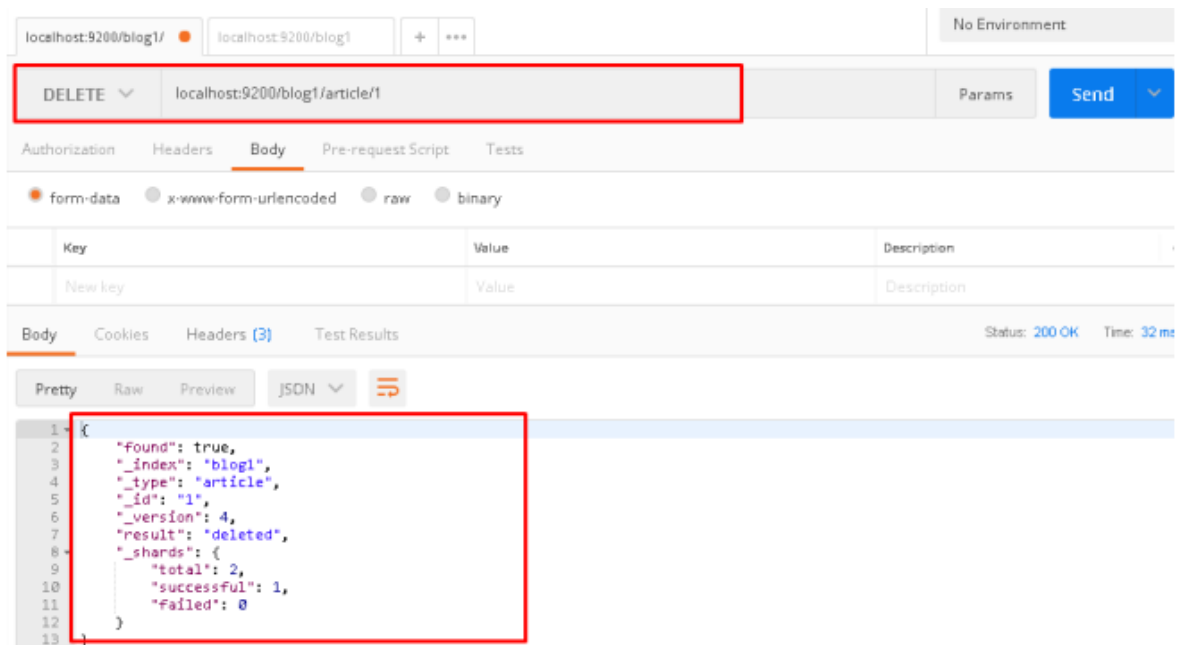


4.4.7 删除文档

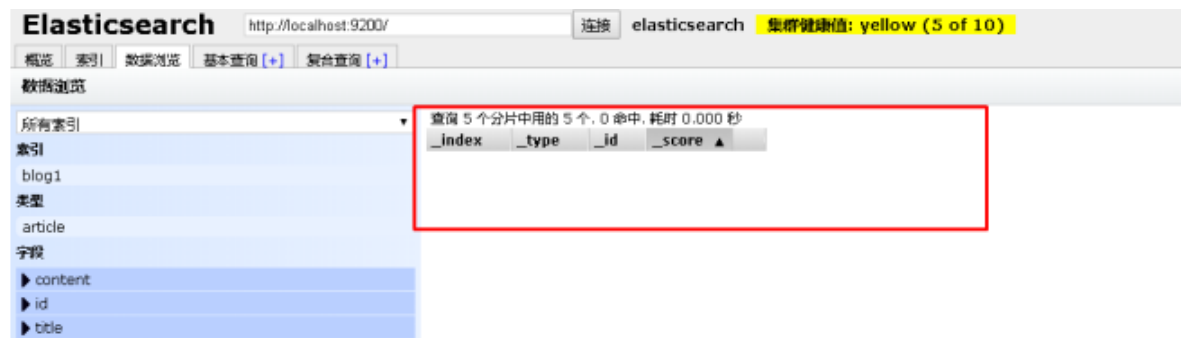
请求方式: DELETE

请求url: `http://localhost:9200/blog1/article/1`

postman截图:



Elasticsearch-head查看:

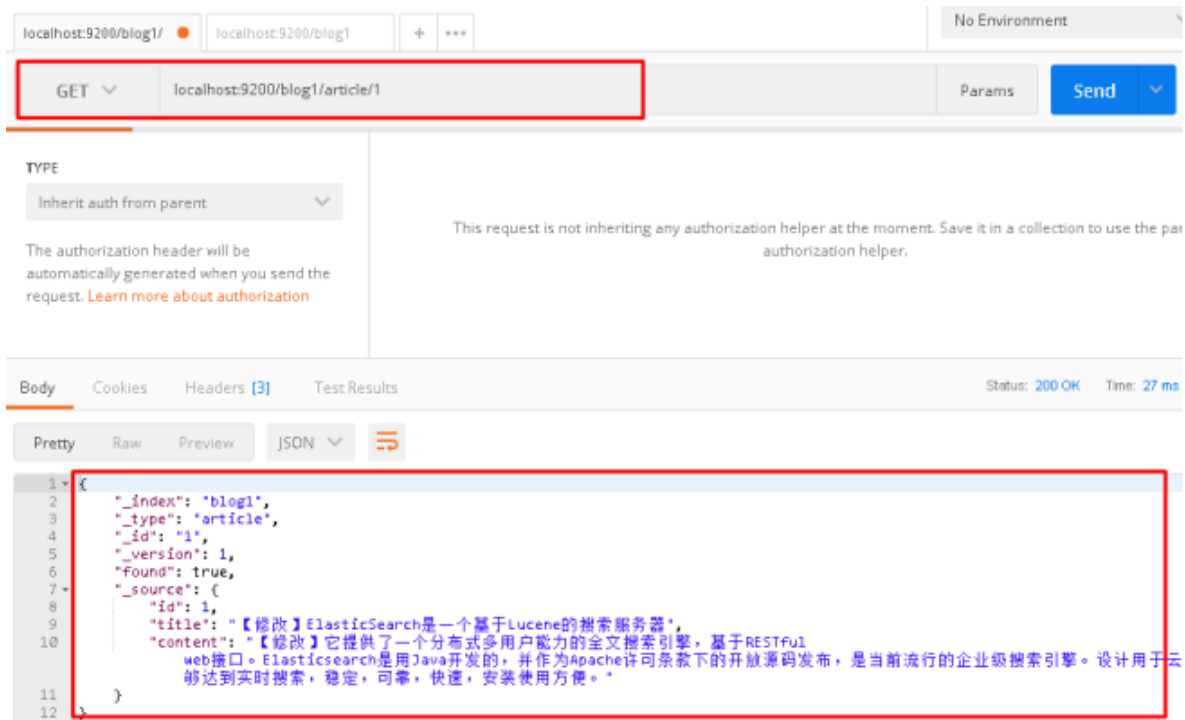


4.4.8 根据id查询

请求方式: GET

请求url: `http://localhost:9200/blog1/article/1`

postman截图:



4.4.9 querystring查询

query_string是带分词器的查询

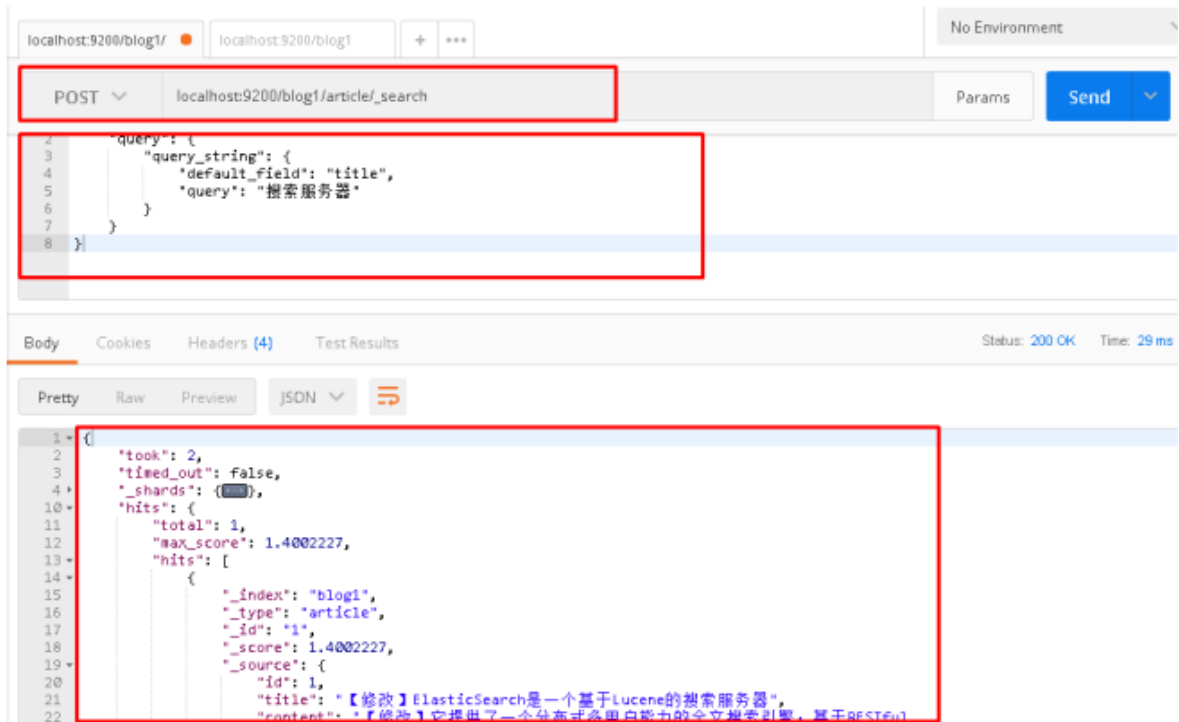
请求方式: POST

请求url: http://localhost:9200/blog1/article/_search

请求体:

```
{  "query": {    "query_string": {      "default_field": "title",      "query": "搜索服务器"    }  }}
```

postman截图:



```
{
  "query": {
    "query_string": {
      "default_field": "title",
      "query": "器"
    }
  }
}
```

4.4.10 term查询

term查询，关键词匹配

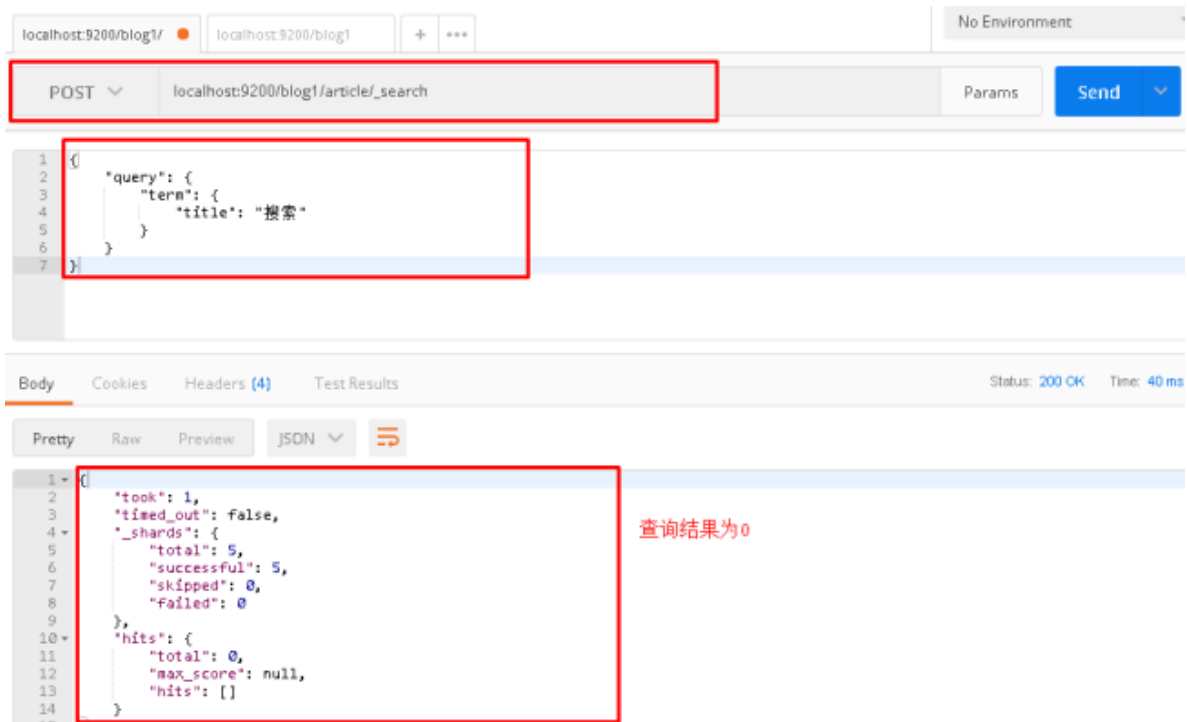
请求url:

请求方式: POST
请求url: `http://localhost:9200/blog1/article/_search`

请求体:

```
{
  "query": {
    "term": {
      "title": "搜索"
    }
  }
}
```

postman截图:



总结:

1. 创建索引库,设置mapping,新建type,删除索引
2. 文档,CRUD
3. 三种查询:

注意: 将搜索内容"搜索服务器"修改为"大器", 同样也能搜索到文档, 该原因会在下面讲解中得到答案

```
{
  "query": {
    "query_string": {
      "default_field": "title",
      "query": "大器"
    }
  }
}
```

第五章 IK 分词器和Elasticsearch集成使用

5.1 上述查询存在问题分析

在进行字符串查询时, 我们发现去搜索"搜索服务器"和"大器"都可以搜索到数据;

而在进行词条查询时, 我们搜索"搜索"却没有搜索到数据;

究其原因是Elasticsearch的标准分词器导致的, 当我们创建索引时, 字段使用的是标准分词器:

```
{
  "mappings": {
    "article": {
```

```

        "properties": {
            "id": {
                "type": "long",
                "store": true,
                "index": "not_analyzed"
            },
            "title": {
                "type": "text",
                "store": true,
                "index": "analyzed",
                "analyzer": "standard" //标准分词器
            },
            "content": {
                "type": "text",
                "store": true,
                "index": "analyzed",
                "analyzer": "standard" //标准分词器
            }
        }
    }
}

```

例如对 "我是程序员" 进行分词

标准分词器分词效果测试：

http://127.0.0.1:9200/_analyze?analyzer=standard&pretty=true&text=我是程序员

分词结果：

```

{
  "tokens" : [
    {
      "token" : "我",
      "start_offset" : 0,
      "end_offset" : 1,
      "type" : "<IDEOGRAPHIC>",
      "position" : 0
    },
    {
      "token" : "是",
      "start_offset" : 1,
      "end_offset" : 2,
      "type" : "<IDEOGRAPHIC>",
      "position" : 1
    },
    {
      "token" : "程",
      "start_offset" : 2,
      "end_offset" : 3,
      "type" : "<IDEOGRAPHIC>",
      "position" : 2
    },
    {
      "token" : "序",
      "start_offset" : 3,

```

```
    "end_offset" : 4,
    "type" : "<IDEOGRAPHIC>",
    "position" : 3
  },
  {
    "token" : "员",
    "start_offset" : 4,
    "end_offset" : 5,
    "type" : "<IDEOGRAPHIC>",
    "position" : 4
  }
]
```

而我们需要的分词效果是：我、是、程序、程序员

这样的话就需要对中文支持良好的分析器的支持，支持中文分词的分词器有很多，word分词器、庖丁解牛、盘古分词、Ansj分词等，但我们常用的还是下面要介绍的IK分词器。

5.2 IK分词器简介

IKAnalyzer是一个开源的，基于java语言开发的轻量级的中文分词工具包。从2006年12月推出1.0版开始，IKAnalyzer已经推出了3个大版本。最初，它是以开源项目Lucene为应用主体的，结合词典分词和文法分析算法的中文分词组件。新版本的IKAnalyzer3.0则发展为面向Java的公用分词组件，独立于Lucene项目，同时提供了对Lucene的默认优化实现。

IK分词器3.0的特性如下：

- 1) 采用了特有的“正向迭代最细粒度切分算法”，具有60万字/秒的高速处理能力。
- 2) 采用了多子处理器分析模式，支持：英文字母（IP地址、Email、URL）、数字（日期，常用中文数量词，罗马数字，科学计数法），中文词汇（姓名、地名处理）等分词处理。
- 3) 对中英联合支持不是很好，在这方面的处理比较麻烦.需再做一次查询，同时是支持个人词条的优化的词典存储，更小的内存占用。
- 4) 支持用户词典扩展定义。
- 5) 针对Lucene全文检索优化的查询分析器IKQueryParser；采用歧义分析算法优化查询关键字的搜索排列组合，能极大的提高Lucene检索的命中率。

5.3 Elasticsearch集成IK分词器

5.3.1 IK分词器的安装

- 1) 下载地址：<https://github.com/medcl/Elasticsearch-analysis-ik/releases>

课程资料也提供了IK分词器的压缩包：



2) 解压, 将解压后的Elasticsearch文件夹拷贝到Elasticsearch-5.6.8\plugins下, 并重命名文件夹为analysis-ik

config	2017/11/15 3:59	文件夹	
commons-codec-1.9.jar	2015/7/2 7:21	Executable Jar File	258 KB
commons-logging-1.2.jar	2015/7/2 7:21	Executable Jar File	61 KB
elasticsearch-analysis-ik-5.6.8.jar	2018/3/5 15:25	Executable Jar File	51 KB
httpclient-4.5.2.jar	2016/8/14 19:32	Executable Jar File	720 KB
httpcore-4.4.4.jar	2016/8/14 19:32	Executable Jar File	320 KB
plugin-descriptor.properties	2018/3/5 15:26	PROPERTIES 文件	3 KB

3) 重新启动Elasticsearch, 即可加载IK分词器

```
Elasticsearch 5.6.8
[2018-04-06T09:21:37,896][INFO][o.e.p.PluginsService][7TD9pkg] loaded module [lang-groovy]
[2018-04-06T09:21:37,896][INFO][o.e.p.PluginsService][7TD9pkg] loaded module [lang-mustache]
[2018-04-06T09:21:37,896][INFO][o.e.p.PluginsService][7TD9pkg] loaded module [lang-painless]
[2018-04-06T09:21:37,896][INFO][o.e.p.PluginsService][7TD9pkg] loaded module [parent-join]
[2018-04-06T09:21:37,896][INFO][o.e.p.PluginsService][7TD9pkg] loaded module [percolator]
[2018-04-06T09:21:37,912][INFO][o.e.p.PluginsService][7TD9pkg] loaded module [reindex]
[2018-04-06T09:21:37,912][INFO][o.e.p.PluginsService][7TD9pkg] loaded module [transport-netty3]
[2018-04-06T09:21:37,912][INFO][o.e.p.PluginsService][7TD9pkg] loaded module [transport-netty4]
[2018-04-06T09:21:37,912][INFO][o.e.p.PluginsService][7TD9pkg] loaded plugin [analysis-ik]
[2018-04-06T09:21:39,215][INFO][o.e.d.DiscoveryModule][7TD9pkg] using discovery type [zen]
[2018-04-06T09:21:39,651][INFO][o.e.n.Node][7TD9pkg] initialized
[2018-04-06T09:21:39,651][INFO][o.e.n.Node][7TD9pkg] starting ...
[2018-04-06T09:21:40,256][INFO][o.e.t.TransportService][7TD9pkg] publish_address {127.0.0.1:9300}, bound_addresses {127.0.0.1:9300}, {[::1]:9300}
[2018-04-06T09:21:43,308][INFO][o.e.c.s.ClusterService][7TD9pkg] new_master {7TD9pkg} {7TD9pkg:SiGYc6dB5JG1TA} {I6zuXWIKTdVfIaPZckkkIQ} {127.0.0.1} {127.0.0.1:9300}, reason: zen-disco-elected-as-master ([0] nodes joined)
[2018-04-06T09:21:43,371][INFO][o.w.a.d.Monitor][7TD9pkg] try load config from C:\elasticsearch-5.6.8\config\analysis-ik\IKAnalyzer.cfg.xml
[2018-04-06T09:21:43,371][INFO][o.w.a.d.Monitor][7TD9pkg] try load config from C:\elasticsearch-5.6.8\plugins\ik\config\IKAnalyzer.cfg.xml
[2018-04-06T09:21:43,697][INFO][o.e.h.n.Netty4HttpServerTransport][7TD9pkg] publish_address {127.0.0.1:9200}, bound_addresses {127.0.0.1:9200}, {[::1]:9200}
[2018-04-06T09:21:43,697][INFO][o.e.n.Node][7TD9pkg] started
[2018-04-06T09:21:43,753][INFO][o.e.g.GatewayService][7TD9pkg] recovered [2] indices into cluster_state
[2018-04-06T09:21:43,959][INFO][o.e.c.r.a.AllocationService][7TD9pkg] Cluster health status changed from [RED] to [YELLOW] (reason: [shards started [[.kibana][0]] ...]).
```

5.3.2 IK分词器测试

IK提供了两个分词算法ik_smart 和 ik_max_word

其中 ik_smart 为最少切分, ik_max_word为最细粒度划分

我们分别来试一下

1) 最小切分: 在浏览器地址栏输入地址

http://127.0.0.1:9200/_analyze?analyzer=ik_smart&pretty=true&text=我是程序员

输出的结果为:

```
{
  "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
  }
]
}

```

2) 最细切分：在浏览器地址栏输入地址

http://127.0.0.1:9200/_analyze?analyzer=ik_max_word&pretty=true&text=我是程序员

输出的结果为：

```

{
  "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
  }
]
}
```

5.4 修改索引映射mapping

5.4.1 重建索引

删除原有blog1索引

请求方式: DELETE
请求地址: <http://localhost:9200/blog1>

创建blog1索引, 此时分词器使用ik_max_word

请求方式: PUT
请求地址: <http://localhost:9200/blog1>

```
{
  "mappings": {
    "article": {
      "properties": {
        "id": {
          "type": "long",
          "store": true,
          "index": "not_analyzed"
        },
        "title": {
          "type": "text",
          "store": true,
          "index": "analyzed",
          "analyzer": "ik_max_word"
        },
        "content": {
          "type": "text",
          "store": true,
          "index": "analyzed",
          "analyzer": "ik_max_word"
        }
      }
    }
  }
}
```

创建文档

请求方式: POST
请求地址: <http://localhost:9200/blog1/article/1>


```
{
  "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。"
}
```

5.4.2 再次测试queryString查询

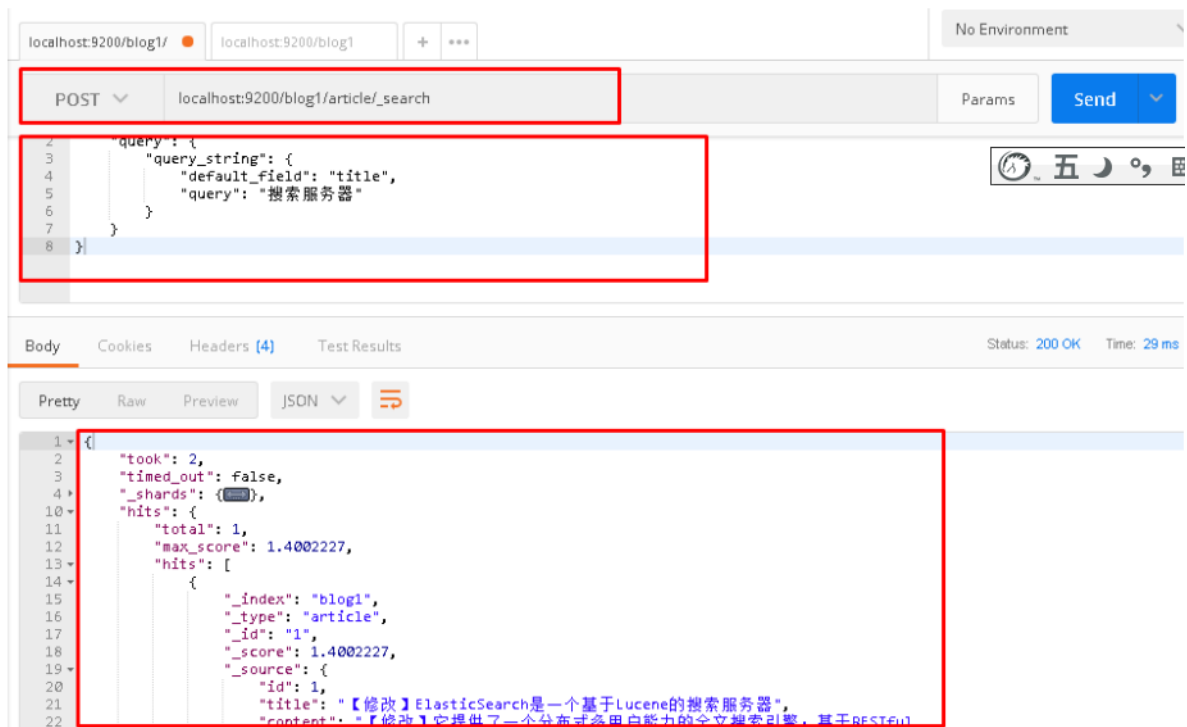
请求url:

请求方式: POST
请求地址: http://localhost:9200/blog1/article/_search

请求体:

```
{
  "query": {
    "query_string": {
      "default_field": "title",
      "query": "搜索服务器"
    }
  }
}
```

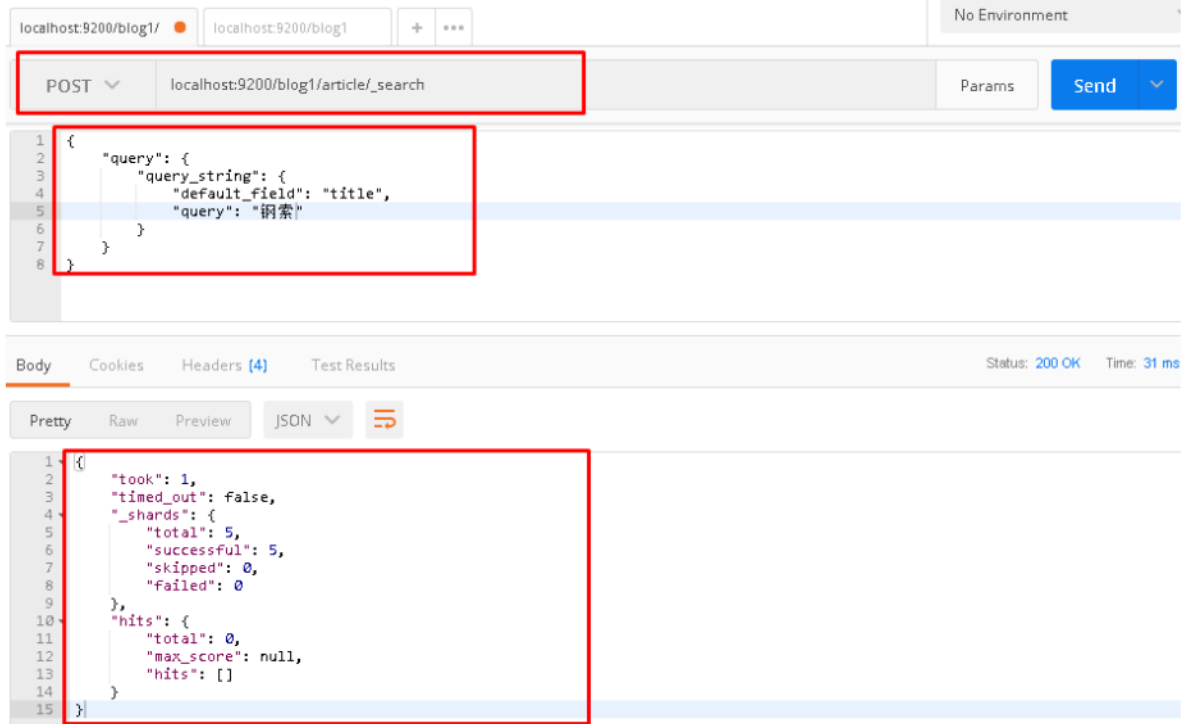
postman截图:



将请求体搜索字符串修改为"大器"，再次查询：

```
{
  "query": {
    "query_string": {
      "default_field": "title",
      "query": "大器"
    }
  }
}
```

postman截图:



5.4.3 再次测试term测试

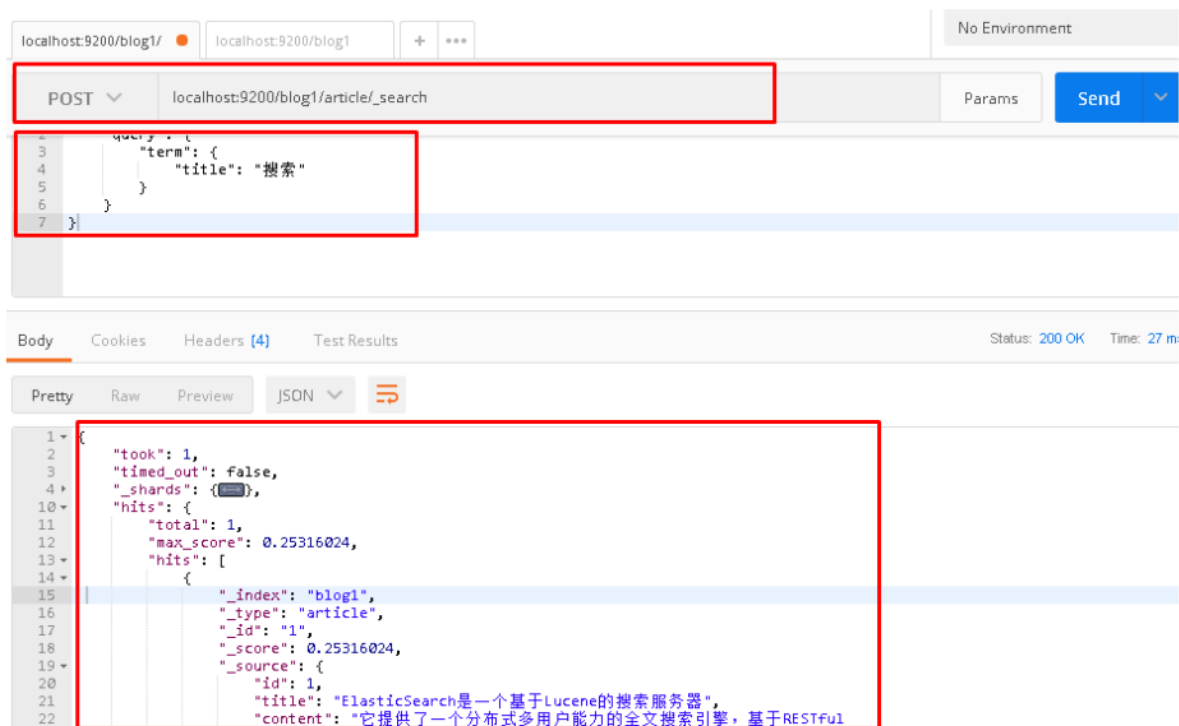
请求url:

请求方式: POST
请求地址: `http://localhost:9200/blog1/article/_search`

请求体:

```
{
  "query": {
    "term": {
      "title": "搜索"
    }
  }
}
```

postman截图:



第六章 Elasticsearch集群

Elasticsearch集群是一个 P2P类型(使用 gossip 协议)的分布式系统，除了集群状态管理以外，其他所有的请求都可以发送到集群内任意一台节点上，这个节点可以自己找到需要转发给哪些节点，并且直接跟这些节点通信。所以，从网络架构及服务配置上来说，构建集群所需要的配置极其简单。在 Elasticsearch 2.0 之前，无阻碍的网络下，所有配置了相同 cluster.name 的节点都自动归属到一个集群中。2.0 版本之后，基于安全的考虑避免开发环境过于随便造成的麻烦，从 2.0 版本开始，默认的自动发现方式改为了单播(unicast)方式。配置里提供几台节点的地址，ES 将其视作 gossip router 角色，借以完成集群的发现。由于这只是 ES 内一个很小的功能，所以 gossip router 角色并不需要单独配置，每个 ES 节点都可以担任。所以，采用单播方式的集群，各节点都配置相同的几个节点列表作为 router 即可。

集群中节点数量没有限制，大于等于2个节点就可以看做是集群了。一般出于高性能及高可用方面来考虑集群中节点数量都是3个以上。

6.1 集群的相关概念

6.1.1 集群 cluster

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

6.1.2 节点 node

一个节点是集群中的一个服务，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。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。

6.1.3 分片和复制 shards&replicas

为了解决索引占用空间过大(1TB以上)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。

为了提高分片高可用，Elasticsearch允许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。

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。

6.2 集群的搭建

6.2.1 准备三台Elasticsearch服务器

创建Elasticsearch-cluster文件夹，在内部复制三个Elasticsearch服务

6.2.2 修改每台服务器配置

修改Elasticsearch-cluster\node*\config\Elasticsearch.yml配置文件

node1节点:

```
#节点1的配置信息:
#集群名称，保证唯一
cluster.name: my-Elasticsearch
#节点名称，必须不一样
node.name: node-1
#必须为本机的ip地址
network.host: 127.0.0.1
#服务端口号，在同一机器下必须不一样
http.port: 9201
#集群间通信端口号，在同一机器下必须不一样
transport.tcp.port: 9301
#设置集群自动发现机器ip集合
discovery.zen.ping.unicast.hosts:
["127.0.0.1:9301", "127.0.0.1:9302", "127.0.0.1:9303"]
```

node2节点:

```
#节点2的配置信息：
#集群名称，保证唯一
cluster.name: my-Elasticsearch
#节点名称，必须不一样
node.name: node-2
#必须为本机的ip地址
network.host: 127.0.0.1
#服务端口号，在同一机器下必须不一样
http.port: 9202
#集群间通信端口号，在同一机器下必须不一样
transport.tcp.port: 9302
#设置集群自动发现机器ip集合
discovery.zen.ping.unicast.hosts:
["127.0.0.1:9301", "127.0.0.1:9302", "127.0.0.1:9303"]
```

node3节点：

```
#节点3的配置信息：
#集群名称，保证唯一
cluster.name: my-Elasticsearch
#节点名称，必须不一样
node.name: node-3
#必须为本机的ip地址
network.host: 127.0.0.1
#服务端口号，在同一机器下必须不一样
http.port: 9203
#集群间通信端口号，在同一机器下必须不一样
transport.tcp.port: 9303
#设置集群自动发现机器ip集合
discovery.zen.ping.unicast.hosts:
["127.0.0.1:9301", "127.0.0.1:9302", "127.0.0.1:9303"]
```

6.2.3 启动各个节点服务器

双击Elasticsearch-cluster\node*\bin\Elasticsearch.bat

启动节点1：

```
Elasticsearch 5.6.8
also, -Dlog4j2.disable.jmx=true, -Dlog4j.skipJansi=true, -XX:+HeapDumpOnOutOfMemoryError, -Delasticsearch, -Des.path.home
e:D:\elasticsearch-cluster\node01]
[2019-04-06T17:42:58,054][INFO ][o.e.p.PluginsService ] [node-1] loaded module [aggs-matrix-stats]
[2019-04-06T17:42:58,054][INFO ][o.e.p.PluginsService ] [node-1] loaded module [ingest-common]
[2019-04-06T17:42:58,054][INFO ][o.e.p.PluginsService ] [node-1] loaded module [lang-expression]
[2019-04-06T17:42:58,054][INFO ][o.e.p.PluginsService ] [node-1] loaded module [lang-groovy]
[2019-04-06T17:42:58,054][INFO ][o.e.p.PluginsService ] [node-1] loaded module [lang-mustache]
[2019-04-06T17:42:58,054][INFO ][o.e.p.PluginsService ] [node-1] loaded module [lang-painless]
[2019-04-06T17:42:58,054][INFO ][o.e.p.PluginsService ] [node-1] loaded module [parent-join]
[2019-04-06T17:42:58,054][INFO ][o.e.p.PluginsService ] [node-1] loaded module [percolator]
[2019-04-06T17:42:58,054][INFO ][o.e.p.PluginsService ] [node-1] loaded module [reindex]
[2019-04-06T17:42:58,054][INFO ][o.e.p.PluginsService ] [node-1] loaded module [transport-netty3]
[2019-04-06T17:42:58,054][INFO ][o.e.p.PluginsService ] [node-1] loaded module [transport-netty4]
[2019-04-06T17:42:58,054][INFO ][o.e.p.PluginsService ] [node-1] no plugins loaded
[2019-04-06T17:43:01,001][INFO ][o.e.d.DiscoveryModule ] [node-1] using discovery type [zen]
[2019-04-06T17:43:01,905][INFO ][o.e.n.Node ] [node-1] initialized
[2019-04-06T17:43:01,906][INFO ][o.e.n.Node ] [node-1] starting ...
[2019-04-06T17:43:02,386][INFO ][o.e.t.TransportService ] [node-1] publish_address {127.0.0.1:9300}, bound_addresses {
127.0.0.1:9300}
[2019-04-06T17:43:05,461][INFO ][o.e.c.s.ClusterService ] [node-1] new_master {node-1} {tX1bVUGUR5Wa7CWx8FH8IA} {8-ifsct
LSDqReetMPz2UAA} (127.0.0.1) [127.0.0.1:9300], reason: zen-disco-elected-as-master ([0] nodes joined)
[2019-04-06T17:43:05,492][INFO ][o.e.h.n.Netty4HttpServerTransport] [node-1] publish_address {127.0.0.1:9200}, bound_add
resses {127.0.0.1:9200}
[2019-04-06T17:43:05,492][INFO ][o.e.n.Node ] [node-1] started
[2019-04-06T17:43:05,675][INFO ][o.e.g.GatewayService ] [node-1] recovered [1] indices into cluster_state
```

启动节点2:

```
Elasticsearch 5.6.8
SweepGC, -XX:CMSInitiatingOccupancyFraction=75, -XX:+UseCMSInitiatingOccupancyOnly, -XX:+AlwaysPreTouch, -Xsslm, -Djava.
awt.headless=true, -Dfile.encoding=UTF-8, -Djna.nosys=true, -Djdk.io.permissionsUseCanonicalPath=true, -Dio.netty.noUnsa
fe=true, -Dio.netty.noKeySetOptimization=true, -Dio.netty.recycler.maxCapacityPerThread=0, -Dlog4j.shutdownHookEnabled=f
alse, -Dlog4j2.disable.jmx=true, -Dlog4j.skipJansi=true, -XX:+HeapDumpOnOutOfMemoryError, -Delasticsearch, -Des.path.hom
e=D:\elasticsearch-cluster\node02]
[2019-04-06T17:44:20,268][INFO][o.e.p.PluginsService][node-2] loaded module [aggs-matrix-stats]
[2019-04-06T17:44:20,269][INFO][o.e.p.PluginsService][node-2] loaded module [ingest-common]
[2019-04-06T17:44:20,270][INFO][o.e.p.PluginsService][node-2] loaded module [lang-expression]
[2019-04-06T17:44:20,270][INFO][o.e.p.PluginsService][node-2] loaded module [lang-groovy]
[2019-04-06T17:44:20,271][INFO][o.e.p.PluginsService][node-2] loaded module [lang-mustache]
[2019-04-06T17:44:20,271][INFO][o.e.p.PluginsService][node-2] loaded module [lang-painless]
[2019-04-06T17:44:20,272][INFO][o.e.p.PluginsService][node-2] loaded module [parent-join]
[2019-04-06T17:44:20,273][INFO][o.e.p.PluginsService][node-2] loaded module [percolator]
[2019-04-06T17:44:20,273][INFO][o.e.p.PluginsService][node-2] loaded module [reindex]
[2019-04-06T17:44:20,273][INFO][o.e.p.PluginsService][node-2] loaded module [transport-netty3]
[2019-04-06T17:44:20,274][INFO][o.e.p.PluginsService][node-2] loaded module [transport-netty4]
[2019-04-06T17:44:20,276][INFO][o.e.p.PluginsService][node-2] no plugins loaded
[2019-04-06T17:44:22,738][INFO][o.e.d.DiscoveryModule][node-2] using discovery type [zen]
[2019-04-06T17:44:23,526][INFO][o.e.n.Node][node-2] initialized
[2019-04-06T17:44:23,526][INFO][o.e.n.Node][node-2] starting ...
[2019-04-06T17:44:23,978][INFO][o.e.t.TransportService][node-2] publish_address {127.0.0.1:9301}, bound_addresses {
127.0.0.1:9301}
[2019-04-06T17:44:27,292][INFO][o.e.c.s.ClusterService][node-2] detected_master {node-1} {tX1bVUGUR5Wa7CWx8FH8IA} {8-
ifscLSDqReetMPPr2UAA} {127.0.0.1} {127.0.0.1:9300}, added {node-1} {tX1bVUGUR5Wa7CWx8FH8IA} {8-ifscLSDqReetMPPr2UAA} {127.0.
0.1} {127.0.0.1:9300}, reason: zen-disco-receive(from master [master {node-1} {tX1bVUGUR5Wa7CWx8FH8IA} {8-ifscLSDqReetMP
r2UAA} {127.0.0.1} {127.0.0.1:9300}] committed version [7])
[2019-04-06T17:44:27,365][INFO][o.e.h.n.Netty4HttpServerTransport][node-2] publish_address {127.0.0.1:9201}, bound_add
resses {127.0.0.1:9201}
[2019-04-06T17:44:27,366][INFO][o.e.n.Node][node-2] started
```

启动节点3:

```
Elasticsearch 5.6.8
awt.headless=true, -Dfile.encoding=UTF-8, -Djna.nosys=true, -Djdk.io.permissionsUseCanonicalPath=true, -Dio.netty.noUnsa
fe=true, -Dio.netty.noKeySetOptimization=true, -Dio.netty.recycler.maxCapacityPerThread=0, -Dlog4j.shutdownHookEnabled=f
alse, -Dlog4j2.disable.jmx=true, -Dlog4j.skipJansi=true, -XX:+HeapDumpOnOutOfMemoryError, -Delasticsearch, -Des.path.hom
e=D:\elasticsearch-cluster\node03]
[2019-04-06T17:44:50,352][INFO][o.e.p.PluginsService][node-3] loaded module [aggs-matrix-stats]
[2019-04-06T17:44:50,353][INFO][o.e.p.PluginsService][node-3] loaded module [ingest-common]
[2019-04-06T17:44:50,353][INFO][o.e.p.PluginsService][node-3] loaded module [lang-expression]
[2019-04-06T17:44:50,353][INFO][o.e.p.PluginsService][node-3] loaded module [lang-groovy]
[2019-04-06T17:44:50,354][INFO][o.e.p.PluginsService][node-3] loaded module [lang-mustache]
[2019-04-06T17:44:50,354][INFO][o.e.p.PluginsService][node-3] loaded module [lang-painless]
[2019-04-06T17:44:50,354][INFO][o.e.p.PluginsService][node-3] loaded module [parent-join]
[2019-04-06T17:44:50,355][INFO][o.e.p.PluginsService][node-3] loaded module [percolator]
[2019-04-06T17:44:50,355][INFO][o.e.p.PluginsService][node-3] loaded module [reindex]
[2019-04-06T17:44:50,356][INFO][o.e.p.PluginsService][node-3] loaded module [transport-netty3]
[2019-04-06T17:44:50,357][INFO][o.e.p.PluginsService][node-3] loaded module [transport-netty4]
[2019-04-06T17:44:50,359][INFO][o.e.p.PluginsService][node-3] no plugins loaded
[2019-04-06T17:44:53,050][INFO][o.e.d.DiscoveryModule][node-3] using discovery type [zen]
[2019-04-06T17:44:53,783][INFO][o.e.n.Node][node-3] initialized
[2019-04-06T17:44:53,783][INFO][o.e.n.Node][node-3] starting ...
[2019-04-06T17:44:54,207][INFO][o.e.t.TransportService][node-3] publish_address {127.0.0.1:9302}, bound_addresses {
127.0.0.1:9302}
[2019-04-06T17:44:57,427][INFO][o.e.c.s.ClusterService][node-3] detected_master {node-1} {tX1bVUGUR5Wa7CWx8FH8IA} {8-
ifscLSDqReetMPPr2UAA} {127.0.0.1} {127.0.0.1:9300}, added {node-2} {UFV7PcPyR2KD5pqwcyStYA} {Ver5cq7aRlunHGGqjEaE5Q} {127.0.
0.1} {127.0.0.1:9301}, {node-1} {tX1bVUGUR5Wa7CWx8FH8IA} {8-ifscLSDqReetMPPr2UAA} {127.0.0.1} {127.0.0.1:9300}, reason: zen-
disco-receive(from master [master {node-1} {tX1bVUGUR5Wa7CWx8FH8IA} {8-ifscLSDqReetMPPr2UAA} {127.0.0.1} {127.0.0.1:9300}] co
mmitted version [16])
[2019-04-06T17:44:57,709][INFO][o.e.h.n.Netty4HttpServerTransport][node-3] publish_address {127.0.0.1:9202}, bound_add
resses {127.0.0.1:9202}
[2019-04-06T17:44:57,711][INFO][o.e.n.Node][node-3] started
```

6.2.4 集群测试

添加索引和映射

请求方式: PUT

请求地址: <http://localhost:9201/blog1>

```
{
  "mappings": {
    "article": {
      "properties": {
        "id": {
          "type": "long",
          "store": true,
```

```

        "index": "not_analyzed"
    },
    "title": {
        "type": "text",
        "store": true,
        "index": "analyzed",
        "analyzer": "standard"
    },
    "content": {
        "type": "text",
        "store": true,
        "index": "analyzed",
        "analyzer": "standard"
    }
}
}
}
}
}

```

添加文档

请求地址: POST
 请求地址: http://localhost:9201/blog1/article/1

```

{
    "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。"
}

```

使用Elasticsearch-header查看集群情况

