

# ElasticSearch第一天

---

## 学习目标：

---

1. 能够理解ElasticSearch的作用
2. 能够安装ElasticSearch服务
3. 能够理解ElasticSearch的相关概念
4. 能够使用Postman发送Restful请求操作ElasticSearch
5. 能够理解分词器的作用
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。

### 1.2 ElasticSearch的使用案例

---

- 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构建挖财自己的日志采集和分析体系

### 1.3 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

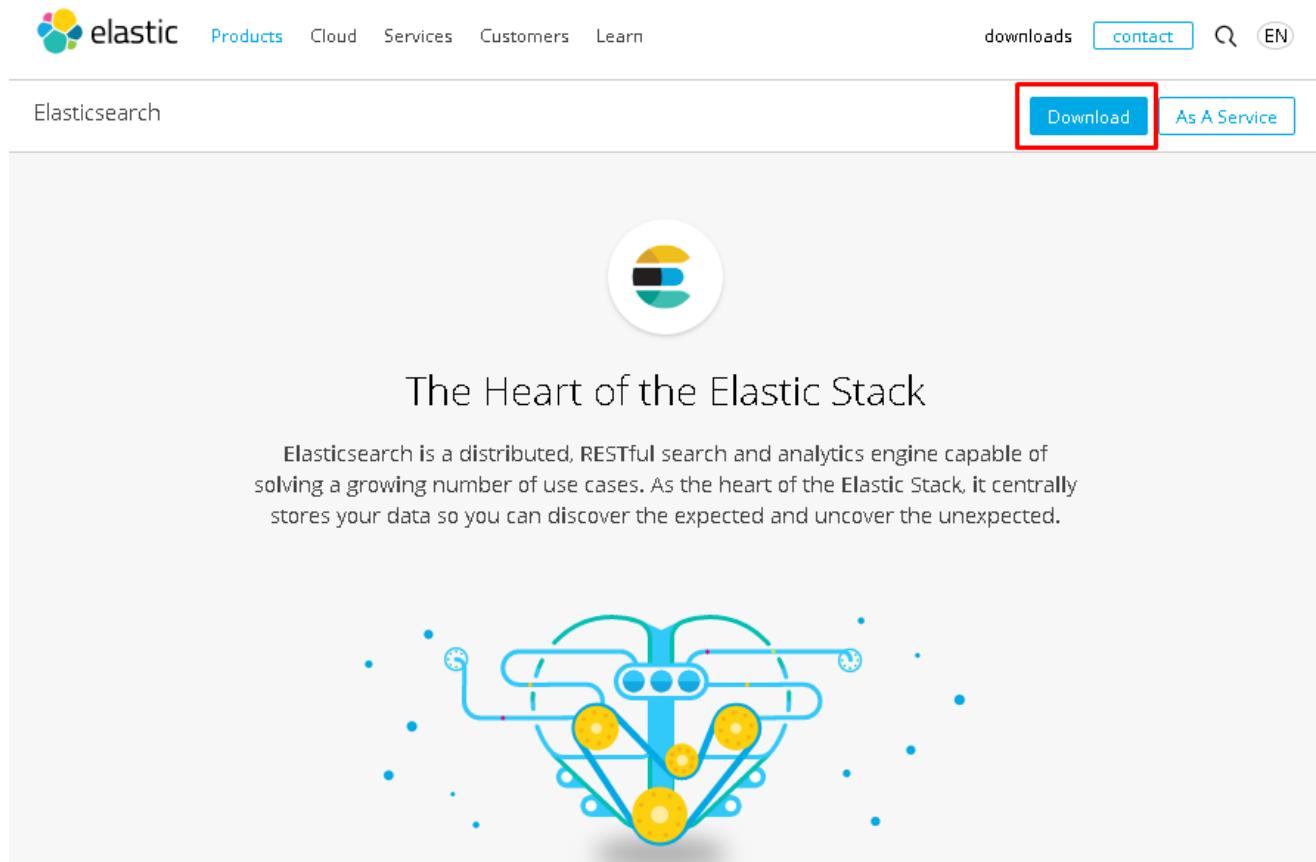
## 第二章 ElasticSearch安装与启动

---

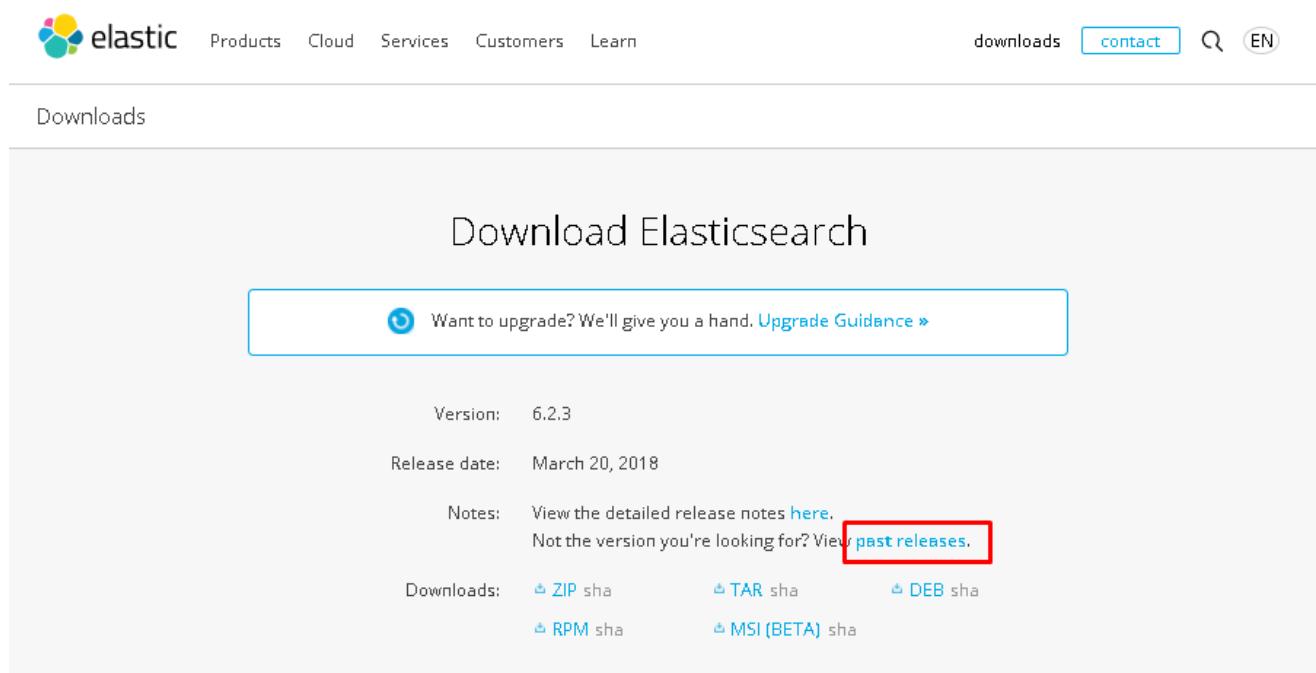
## 2.1 下载ES压缩包

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/products/elasticsearch>



The screenshot shows the official ElasticSearch website. At the top, there is a navigation bar with links for 'Products', 'Cloud', 'Services', 'Customers', 'Learn', 'downloads', 'contact', a search icon, and a language switch to 'EN'. Below the navigation, the word 'Elasticsearch' is displayed. To the right of 'Elasticsearch' are two buttons: 'Download' (which is highlighted with a red box) and 'As A Service'. The main content area features a large circular logo with a stylized 'E' and the text 'The Heart of the Elastic Stack'. Below this, a descriptive paragraph reads: 'Elasticsearch is a distributed, RESTful search and analytics engine capable of solving a growing number of use cases. As the heart of the Elastic Stack, it centrally stores your data so you can discover the expected and uncover the unexpected.' Below the text is a complex, abstract illustration of interconnected nodes and lines in blue and yellow.



The screenshot shows the 'Downloads' page of the ElasticSearch website. At the top, there is a navigation bar with links for 'Products', 'Cloud', 'Services', 'Customers', 'Learn', 'downloads', 'contact', a search icon, and a language switch to 'EN'. Below the navigation, the word 'Downloads' is displayed. The main content area features a large heading 'Download Elasticsearch'. Below this, there is a button with a circular icon and the text 'Want to upgrade? We'll give you a hand. [Upgrade Guidance »](#)'. Underneath, there is a section for version 6.2.3, which includes the 'Version: 6.2.3', 'Release date: March 20, 2018', and 'Notes: View the detailed release notes [here](#). Not the version you're looking for? View [past releases](#)'. At the bottom, there are download links for 'ZIP sha', 'TAR sha', 'DEB sha', 'RPM sha', and 'MSI (BETA) sha'.

## Elasticsearch 6.1.4

[Download](#)

March 20, 2018

## Elasticsearch 6.2.2

[▶ See Release Notes](#)[Download](#)

February 20, 2018

## Elasticsearch 5.6.8

[Download](#)

February 20, 2018



elastic

[Products](#) [Cloud](#) [Services](#) [Customers](#) [Learn](#)[downloads](#)[contact](#)[EN](#)

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

修改elasticsearch配置文件：config/elasticsearch.yml，增加以下两句命令：

```
http.cors.enabled: true
http.cors.allow-origin: "*"
```

此步为允许elasticsearch跨越访问，如果不安装后面的elasticsearch-head是可以不修改，直接启动。

## 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] [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]
[2018-04-04T14:02:41,546][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] [7TD9pkg] initialized
[2018-04-04T14:02:43,262][INFO ][o.e.n.Node] [7TD9pkg] 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]}, {[::1]:9300}
[2018-04-04T14:02:46,996][INFO ][o.e.c.s.ClusterService] [7TD9pkg] new_master {7TD9pkgqSfGYc6dB5JG1TA} {7pCQK
EszTXq38wDGroj4fg} [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]}, {[::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TCPClient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：



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

ElasticSearch不同于Solr自带图形化界面，我们可以通过安装ElasticSearch的head插件，完成图形化界面的效果，完成索引数据的查看。安装插件的方式有两种，在线安装和本地安装。本文档采用本地安装方式进行head插件的安装。elasticsearch-5-\*以上版本安装head需要安装node和grunt

1 ) 下载head插件：<https://github.com/mobz/elasticsearch-head>

在资料中已经提供了elasticsearch-head-master插件压缩包：



elasticsearch-head-master.zip

2 ) 将elasticsearch-head-master压缩包解压到任意目录 , 但是要和elasticsearch的安装目录区别开

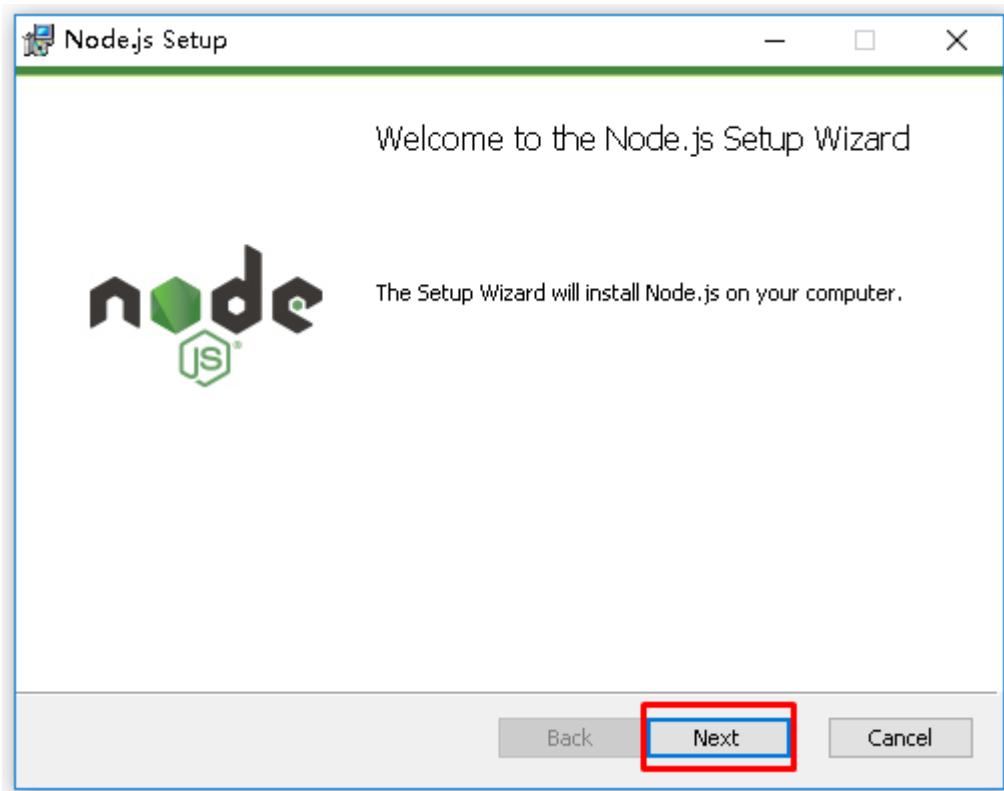
3 ) 下载nodejs : <https://nodejs.org/en/download/>

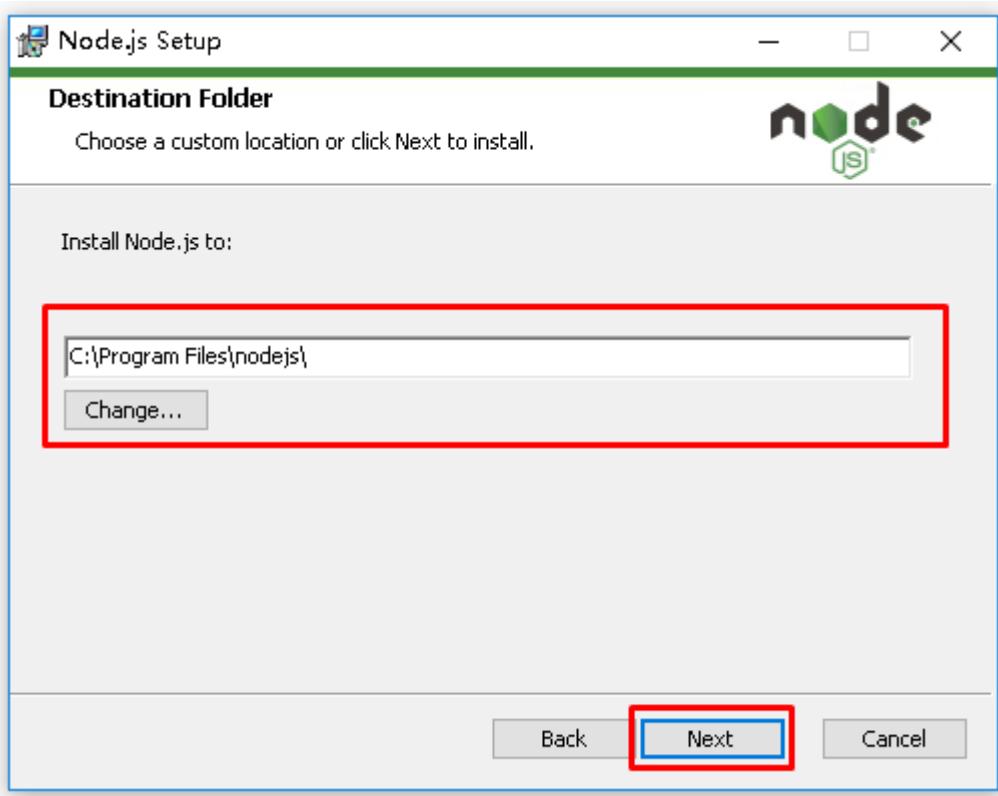
在资料中已经提供了nodejs安装程序 :

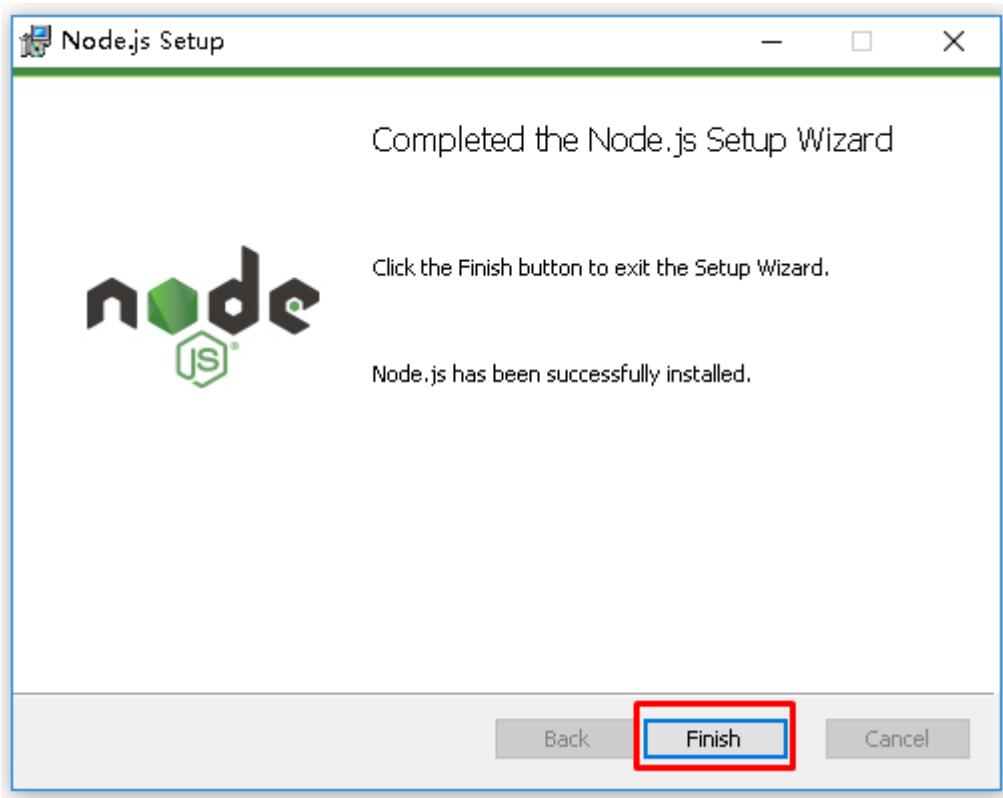
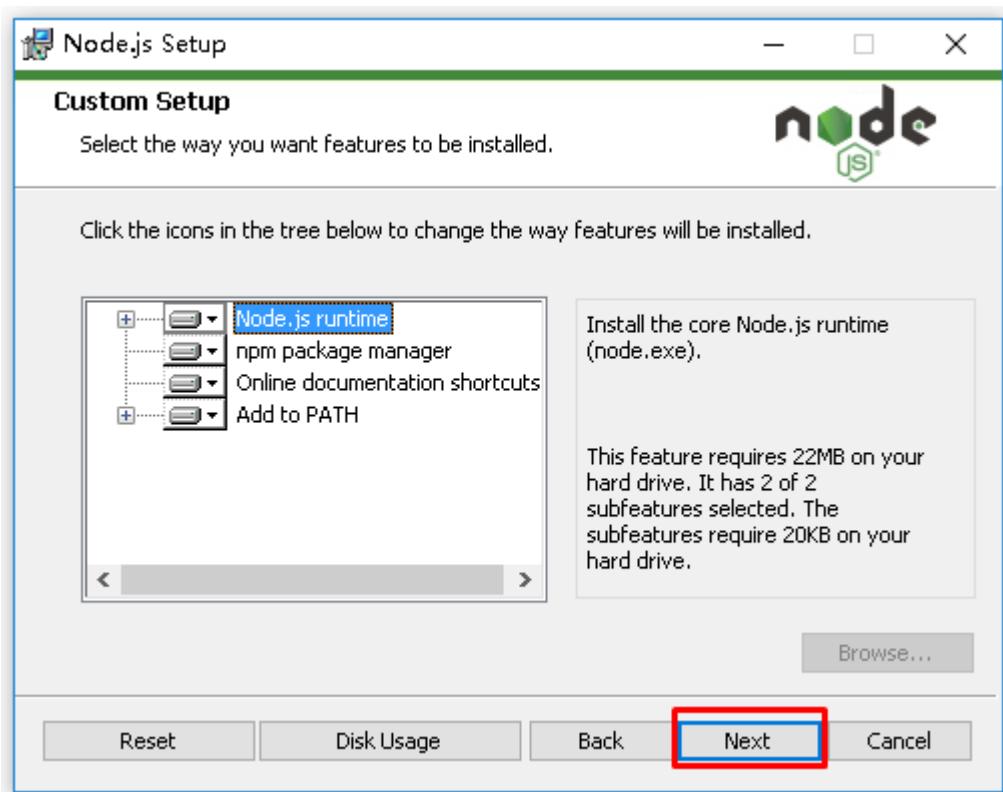


node-v8.9.4-x64.msi

双击安装程序 , 步骤截图如下 :







安装完毕，可以通过cmd控制台输入：node -v 查看版本号

5 ) 将grunt安装为全局命令，Grunt是基于Node.js的项目构建工具

在cmd控制台中输入如下执行命令：

```
npm install -g grunt-cli
```

执行结果如下图：

```
命令提示符
Microsoft Windows [版本 10.0.16299.309]
(c) 2017 Microsoft Corporation。保留所有权利。

C:\Users\muzimoo>npm install -g grunt-cli
C:\Users\muzimoo\AppData\Roaming\npm\grunt -> C:\Users\muzimoo\AppData\Roaming\npm\node_modules\grunt-cli\bin\grunt
+ grunt-cli@1.2.0
added 16 packages in 2.578s

C:\Users\muzimoo>
```

6 ) 进入elasticsearch-head-master目录启动head , 在命令提示符下输入命令：

```
>npm install
>grunt server
```

```
命令提示符
Microsoft Windows [版本 10.0.16299.309]
(c) 2017 Microsoft Corporation。保留所有权利。

C:\Users\muzimoo>cd C:\elasticsearch-head-master

C:\elasticsearch-head-master>grunt server
(node:12764) ExperimentalWarning: The http2 module is an experimental API.
Running "connect:server" (connect) task
Waiting forever...
Started connect web server on http://localhost:9100
```

7 ) 打开浏览器 , 输入 <http://localhost:9100> , 看到如下页面：



如果不能成功连接到es服务 , 需要修改ElasticSearch的config目录下的配置文件 : config/elasticsearch.yml , 增加以下两句命令 :

```
http.cors.enabled: true
http.cors.allow-origin: "*"
```

然后重新启动ElasticSearch服务。

## 第三章 Elasticsearch相关概念(术语)

### 3.1 概述

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
```

## 3.2 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

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。

## 第四章 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

- 第三种，使用elasticsearch提供的API进行访问

## 4.1 安裝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 & HTTP 请求调试。软件功能非常强大，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..)，且可以附带任何数量的参数。

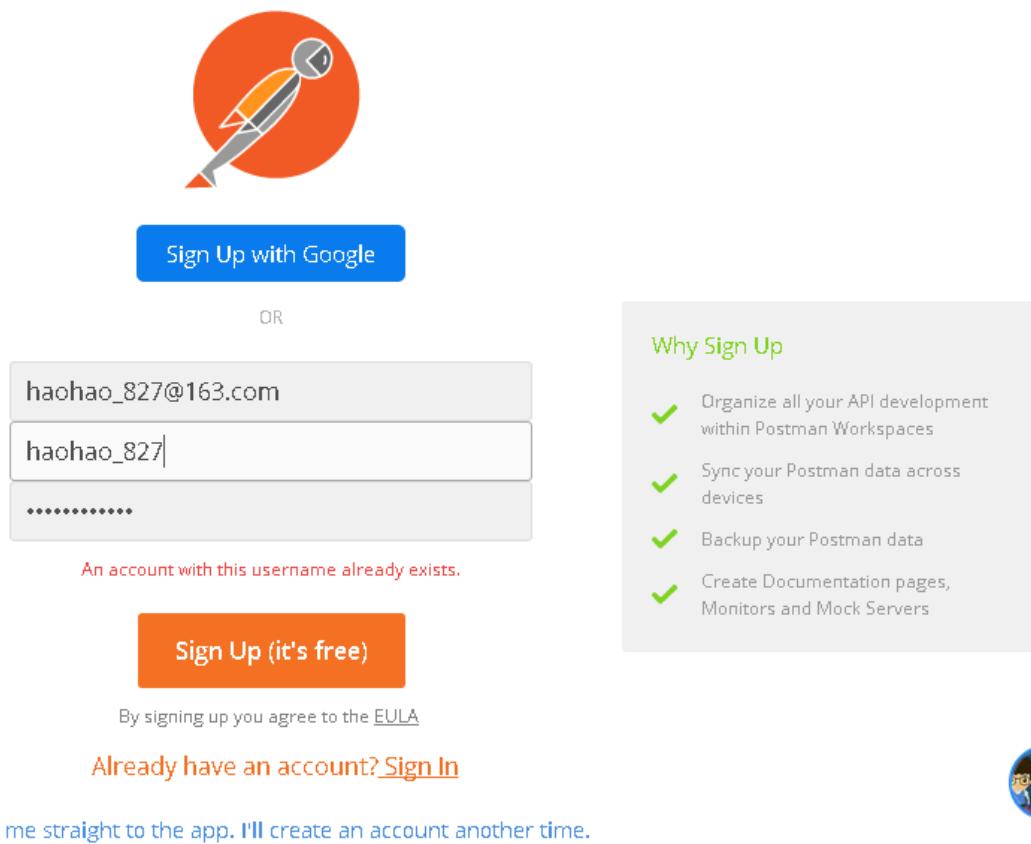
## 4.1 下载Postman工具

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

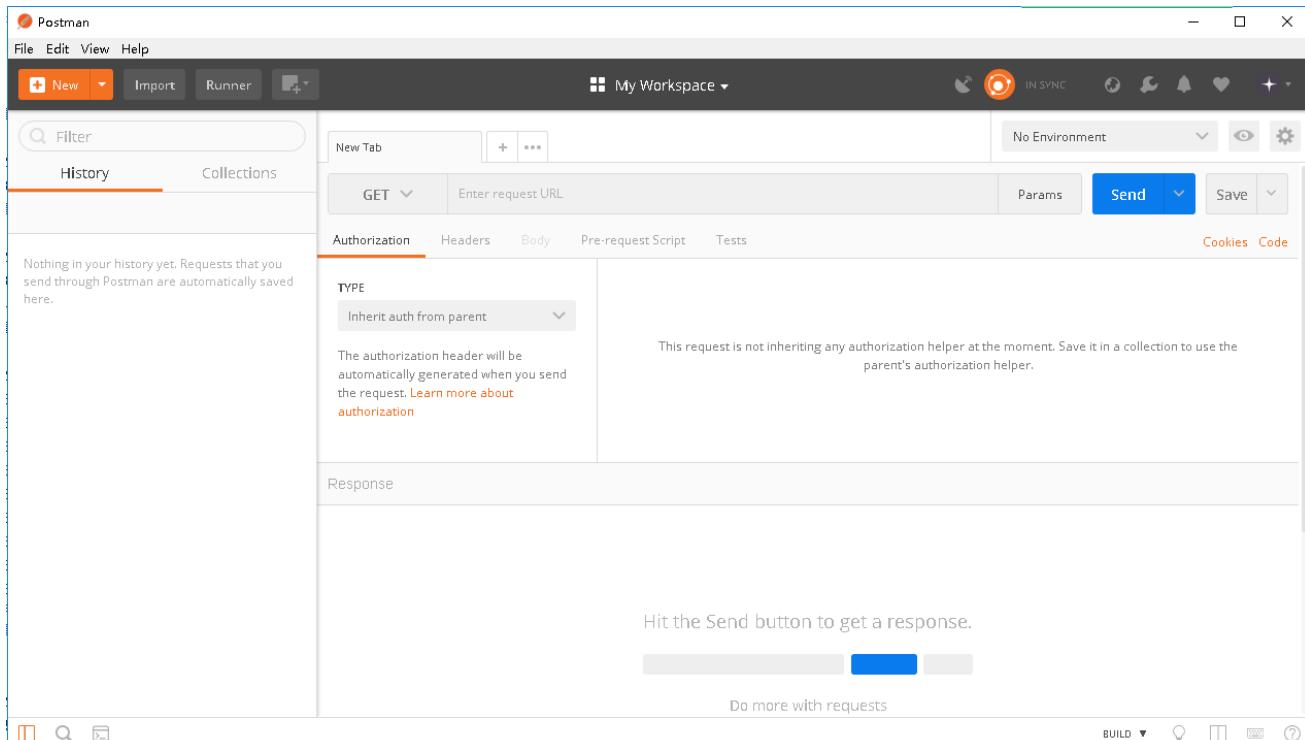
课程资料中已经提供了安装包



## 4.2 注册Postman工具



The image shows the Postman sign-up screen. It features a large orange Postman logo at the top. Below it is a blue button labeled "Sign Up with Google". Underneath that is a "Sign Up" button. To the left of the "Sign Up" button is a "Sign In" button. The main input fields are for "Email" (containing "haohao\_827@163.com") and "Password" (containing "haohao\_827"). Below the password field is a red error message: "An account with this username already exists." To the right of the input fields is a "Why Sign Up" section with a list of benefits: "Organize all your API development within Postman Workspaces", "Sync your Postman data across devices", "Backup your Postman data", and "Create Documentation pages, Monitors and Mock Servers". At the bottom of the screen, there is a note: "By signing up you agree to the [EULA](#)". Below that is a "Sign In" link and a "Take me straight to the app. I'll create an account another time." link. There is also a small circular profile picture in the bottom right corner.



## 4.2 使用Postman工具进行Restful接口访问

### 4.2.1 ElasticSearch的接口语法

```
curl -X<VERB> '<PROTOCOL>://<HOST>:<PORT>/<PATH>?<QUERY_STRING>' -d '<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	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)

### 4.2.2 创建索引index和映射mapping

请求url：

PUT      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：

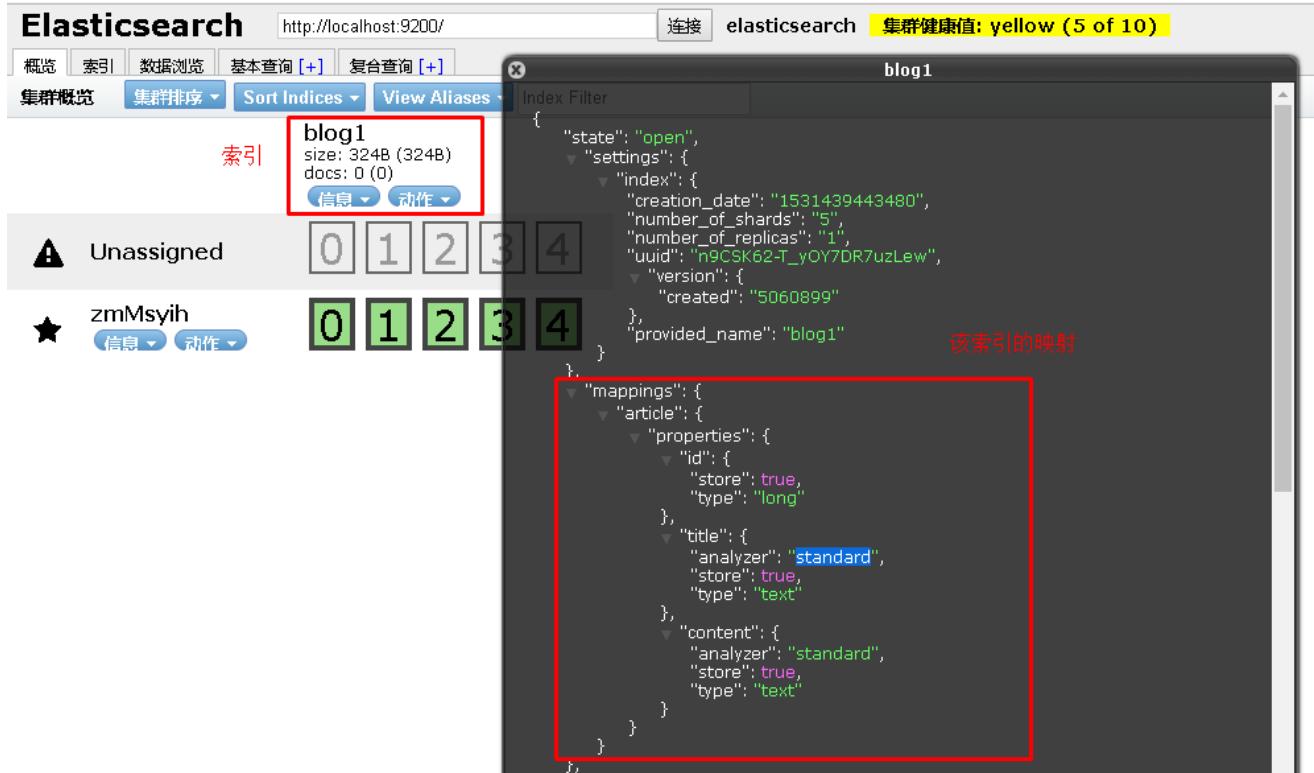
请求url: localhost:9200/blog1 | 请求数量: 1 | Send

Authorization | Headers (1) | Body (raw) | Pre-request Script | Tests

form-data | x-www-form-urlencoded | raw (selected) | binary | Text

1 {  
2 "mappings": {  
3 "article": {  
4 "properties": {  
5 "id": {  
6 "type": "long",  
7 "store": true,  
8 "index": "not\_analyzed"  
9 },  
10 "title": {  
11 "type": "text",  
12 "store": true,  
13 "index": "analyzed",  
14 "analyzer": "standard"  
15 }  
16 }  
17 }  
18 }  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839

elasticsearch-head查看：



The screenshot shows the Elasticsearch-head interface with the 'blog1' index selected. The 'mappings' section is highlighted with a red box. The 'mappings' section contains the following JSON:

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

### 4.2.3 创建索引后设置Mapping

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maping信息，直接使用put方法创建一个索引，然后设置mapping信息。

请求的url：

```
POST    http://127.0.0.1:9200/blog2/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  
            }  
        }  
    }  
}
```

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

PostMan截图

Postman screenshot showing a successful POST request to `http://127.0.0.1:9200/blog2/hello/_mapping`. The request body is a JSON object defining the mapping for the 'hello' index. The response body shows a successful acknowledgment.

**Request Body:**

```
1 {  
2   "hello": {  
3     "properties": {  
4       "id": {  
5         "type": "long",  
6         "store": true  
7       },  
8       "title": {  
9         "type": "text",  
10        "store": true,  
11        "index": true,  
12        "analyzer": "standard"  
13     },  
14     "content": {  
15       "type": "text",  
16       "store": true,  
17       "index": true,  
18     }  
19   }  
20 }
```

**Response Body:**

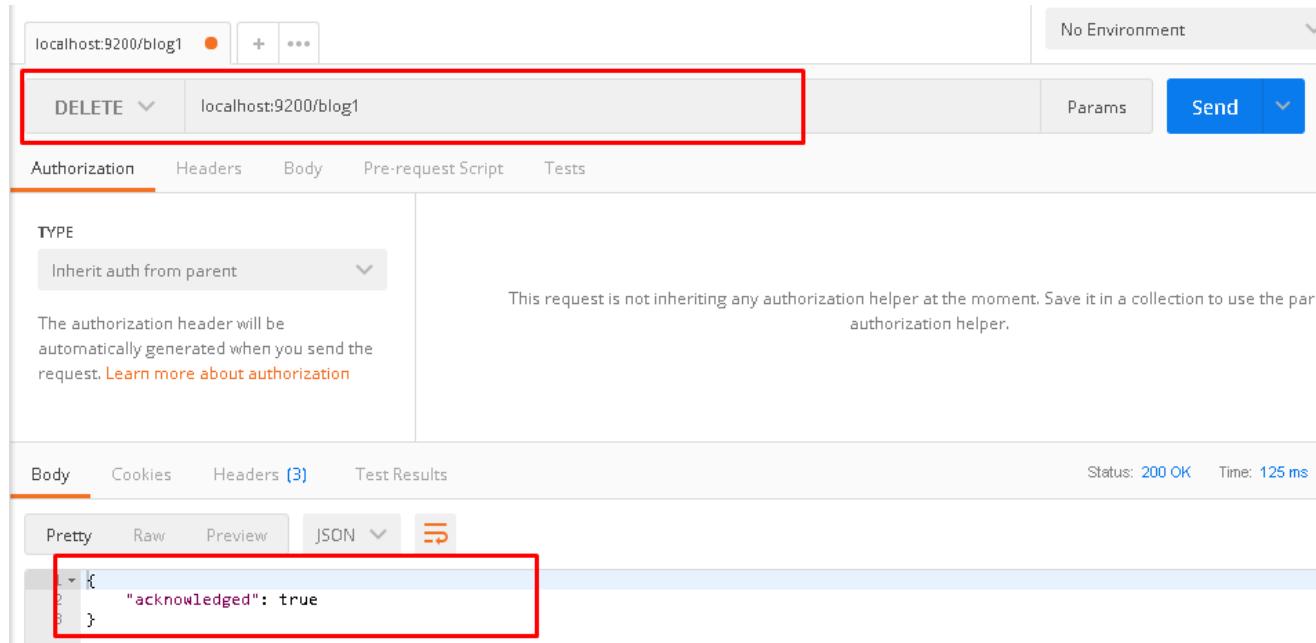
```
1 {  
2   "acknowledged": true  
3 }
```

#### 4.2.4 删除索引index

请求url :

```
DELETE      localhost:9200/blog1
```

postman截图：



Postman screenshot showing a successful DELETE request to `localhost:9200/blog1`. The response body is a JSON object with the following content:

```
{ "acknowledged": true }
```

elasticsearch-head查看：



Elasticsearch-head browser extension screenshot showing the cluster health status as green. The URL is `localhost:9100`.

## 4.2.5 创建文档document

请求url：

```
POST 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：

Postman screenshot showing a POST request to `localhost:9200/blog1/article/1`. The request body is a JSON document:

```
1 {
2     "id":1,
3     "title":"ElasticSearch是一个基于Lucene的搜索服务器",
4     "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
5     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
6     实时搜索，稳定，可靠，快速，安装使用方便。"
7 }
```

The response status is 201 Created. The response body is:

```
1 {
2     "_index": "blog1",
3     "_type": "article",
4     "_id": "1",
5     "_version": 1,
6     "result": "created",
7     "_shards": {
8         "total": 2,
9         "successful": 1,
10        "failed": 0
11     }
12 }
```

elasticsearch-head查看：

Elasticsearch Head plugin screenshot showing the 'Data Browser' tab. A search query is run on the 'blog1' index, returning the following results:

index	_type	_id	_score	id	title	content
blog1	article	1	1	1	ElasticSearch是一个基于Lucene的搜索服务器	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 实时搜索，稳定，可靠，快速，安装使用方便。

## 4.2.6 修改文档document

请求url：

```
POST      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：

POST localhost:9200/blog1/article/1

Body (raw JSON)

```

1 {
2   "id":1,
3   "title": "【修改】ElasticSearch是一个基于Lucene的搜索服务器",
4   "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
5     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
6     实时搜索，稳定，可靠，快速，安装使用方便。"
7 }
  
```

Body (Pretty)

```

1 {
2   "_index": "blog1",
3   "_type": "article",
4   "_id": "1",
5   "_version": 3,
6   "result": "updated",
7   "_shards": {
8     "total": 2,
9     "successful": 1,
10    "failed": 0
  
```

elasticsearch-head查看：

Elasticsearch (http://localhost:9200/)

集群健康值: yellow (5 of 10)

数据浏览器

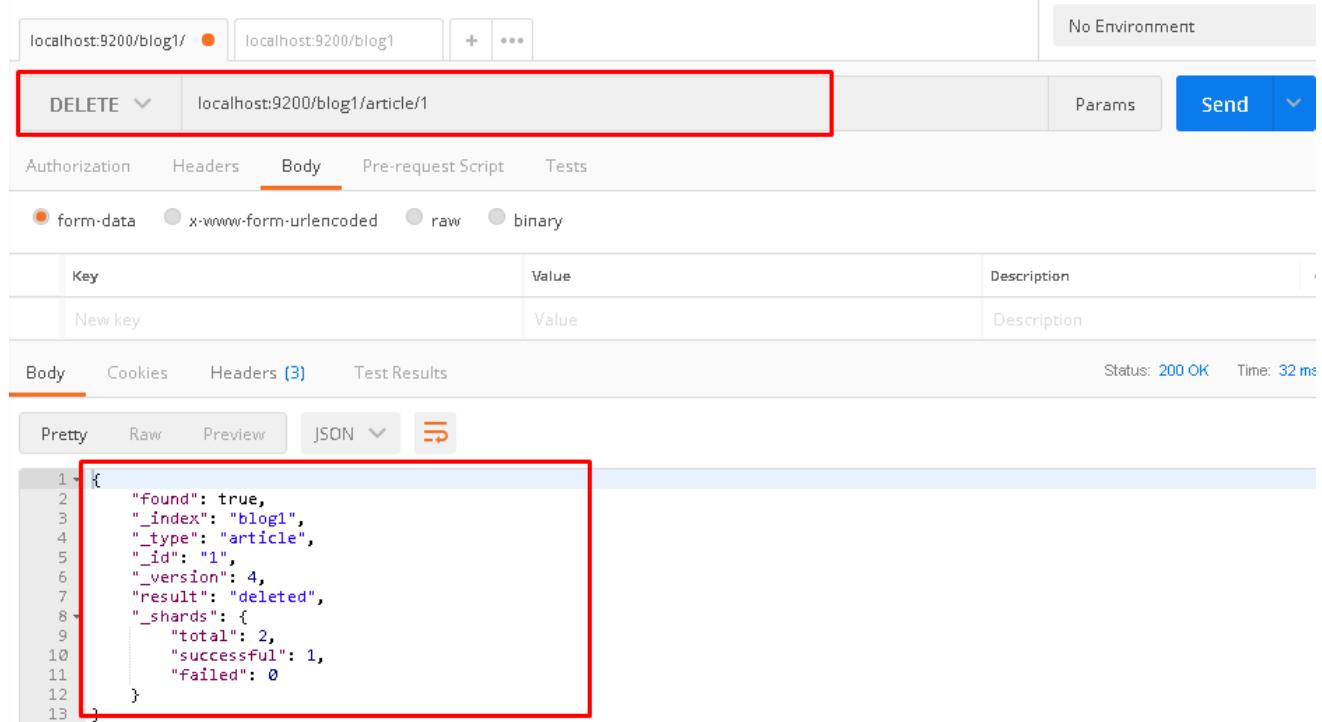
index	type	id	score	id	title	content
blog1	article	1	1	1	【修改】ElasticSearch是一个基于Lucene的搜索服务器	【修改】它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful

## 4.2.7 删除文档document

请求url：

```
DELETE localhost:9200/blog1/article/1
```

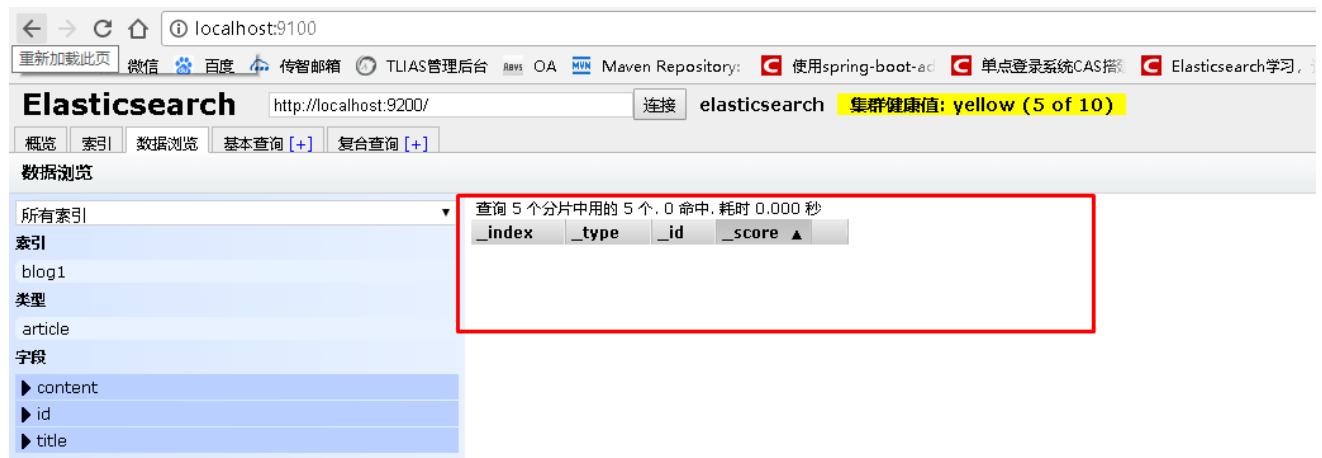
postman截图：



Postman screenshot showing a successful DELETE request to `localhost:9200/blog1/article/1`. The response body is a JSON object indicating the deletion of the document with `_id 1`.

```
1 {  
2   "found": true,  
3   "_index": "blog1",  
4   "_type": "article",  
5   "_id": "1",  
6   "_version": 4,  
7   "result": "deleted",  
8   "_shards": {  
9     "total": 2,  
10    "successful": 1,  
11    "failed": 0  
12  }  
13 }
```

elasticsearch-head查看：



Elasticsearch Head plugin screenshot showing the status of the `blog1` index. The status is yellow (5 of 10 shards healthy).

所有索引 索引 数据浏览 基本查询 [+] 复合查询 [+]

数据浏览

所有索引 索引 blog1 类型 article 字段 content id title

查询 5 个分片中用的 5 个, 0 命中, 耗时 0.000 秒

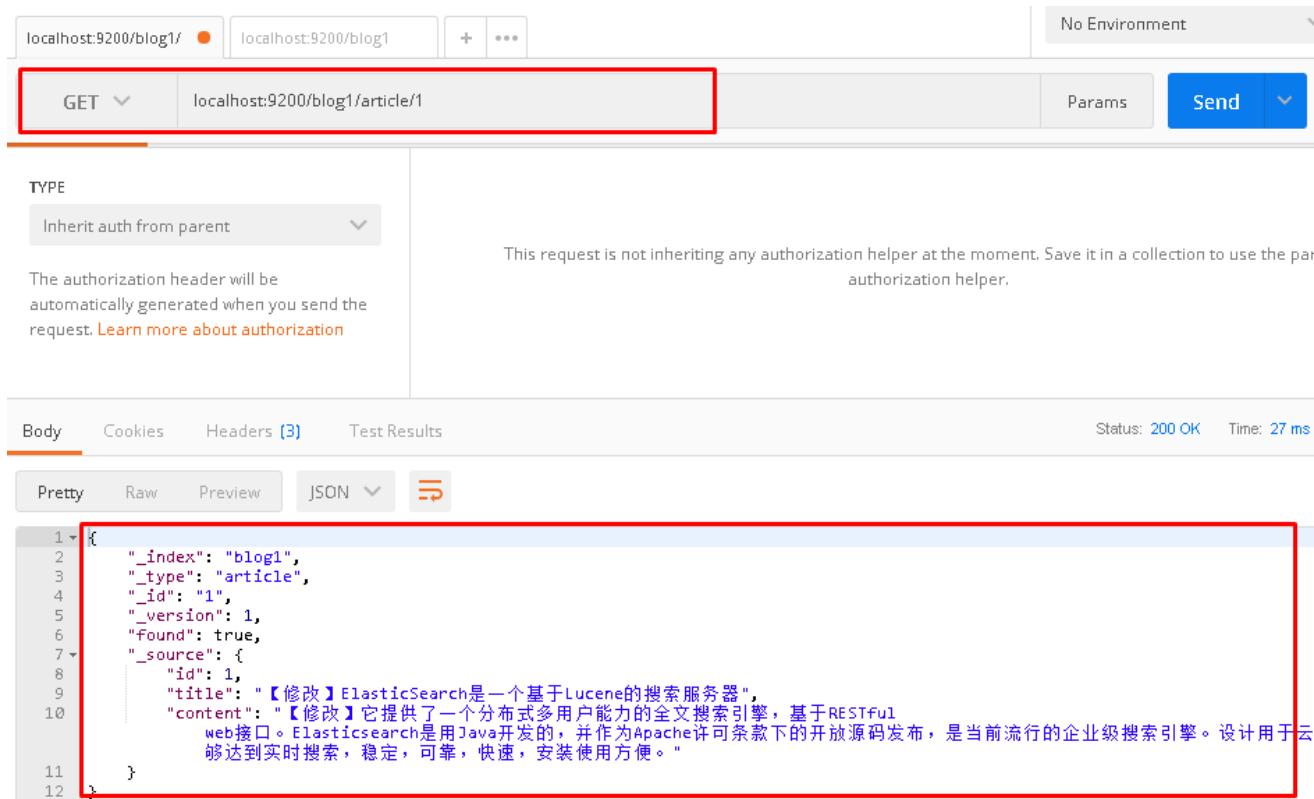
<code>_index</code>	<code>_type</code>	<code>_id</code>	<code>_score</code>

## 4.2.8 查询文档-根据id查询

请求url：

```
GET localhost:9200/blog1/article/1
```

postman截图：



localhost:9200/blog1/ localhost:9200/blog1 + ... No Environment

GET localhost:9200/blog1/article/1 Params Send

TYPE  
Inherit auth from parent

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent authorization helper.

Body Cookies Headers (3) Test Results Status: 200 OK Time: 27 ms

Pretty Raw Preview JSON

```
1 {  
2   "_index": "blog1",  
3   "_type": "article",  
4   "_id": "1",  
5   "_version": 1,  
6   "found": true,  
7   "_source": {  
8     "id": 1,  
9     "title": "【修改】ElasticSearch是一个基于Lucene的搜索服务器",  
10    "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  
11      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  
12      能够达到实时搜索，稳定，可靠，快速，安装使用方便。"  
13  }  
14}
```

## 4.2.9 查询文档-queryString查询

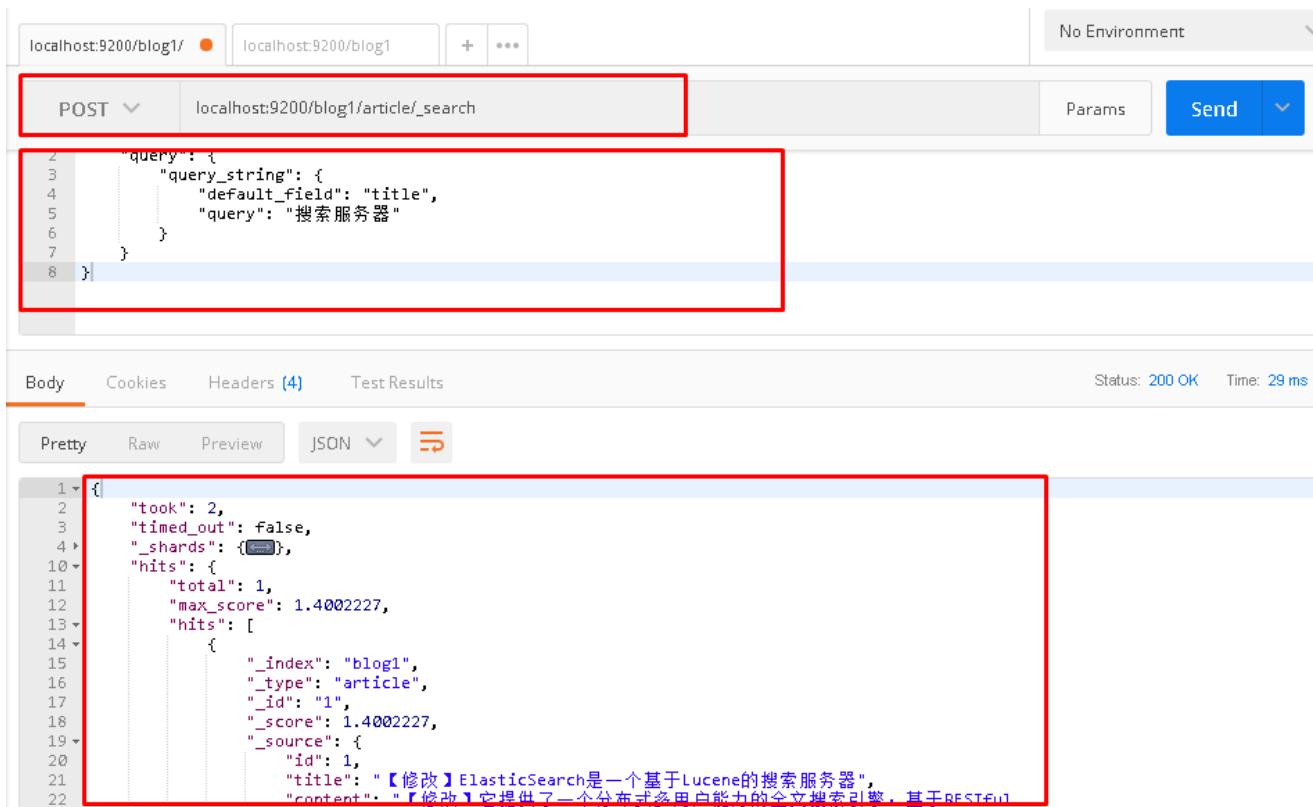
请求url：

```
POST localhost:9200/blog1/article/_search
```

请求体：

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

postman截图：



localhost:9200/blog1/ localhost:9200/blog1/+ ... No Environment

POST localhost:9200/blog1/article/\_search Params Send

```
2 "query": {  
3     "query_string": {  
4         "default_field": "title",  
5         "query": "搜索服务器"  
6     }  
7 }  
8 }
```

Body Cookies Headers (4) Test Results Status: 200 OK Time: 29 ms

Pretty Raw Preview JSON

```
1 {  
2     "took": 2,  
3     "timed_out": false,  
4     "_shards": {  
5     },  
6     "hits": {  
7         "total": 1,  
8         "max_score": 1.4002227,  
9         "hits": [  
10             {  
11                 "_index": "blog1",  
12                 "_type": "article",  
13                 "_id": "1",  
14                 "_score": 1.4002227,  
15                 "_source": {  
16                     "id": 1,  
17                     "title": "【修改】ElasticSearch是一个基于Lucene的搜索服务器",  
18                     "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  
19                 },  
20             }  
21         }  
22     }  
23 }
```

注意：

将搜索内容"搜索服务器"修改为"钢索"，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": "钢索"  
        }  
    }  
}
```

## 4.2.10 查询文档-term查询

请求url：

```
POST localhost:9200/blog1/article/_search
```

请求体：

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

postman截图：

The screenshot shows a Postman request for a POST to `localhost:9200/blog1/article/_search`. The request body contains a JSON query with a red box around it. The response body is also highlighted with a red box and shows a search result with 0 hits, with the text '查询结果为0' (Search result is 0) overlaid.

POST `localhost:9200/blog1/article/_search`

```
1 {  
2   "query": {  
3     "term": {  
4       "title": "搜索"  
5     }  
6   }  
7 }
```

Body Cookies Headers (4) Test Results Status: 200 OK Time: 40 ms

Pretty Raw Preview JSON

```
1 {  
2   "took": 1,  
3   "timed_out": false,  
4   "_shards": {  
5     "total": 5,  
6     "successful": 5,  
7     "skipped": 0,  
8     "failed": 0  
9   },  
10  "hits": {  
11    "total": 0,  
12    "max_score": null,  
13    "hits": []  
14  }  
15 }
```

查询结果为0

## 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"钢索"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": false  
        }  
      }  
    }  
  }  
}
```

```
        "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](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
    }
  ]
}
```

```
        "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：

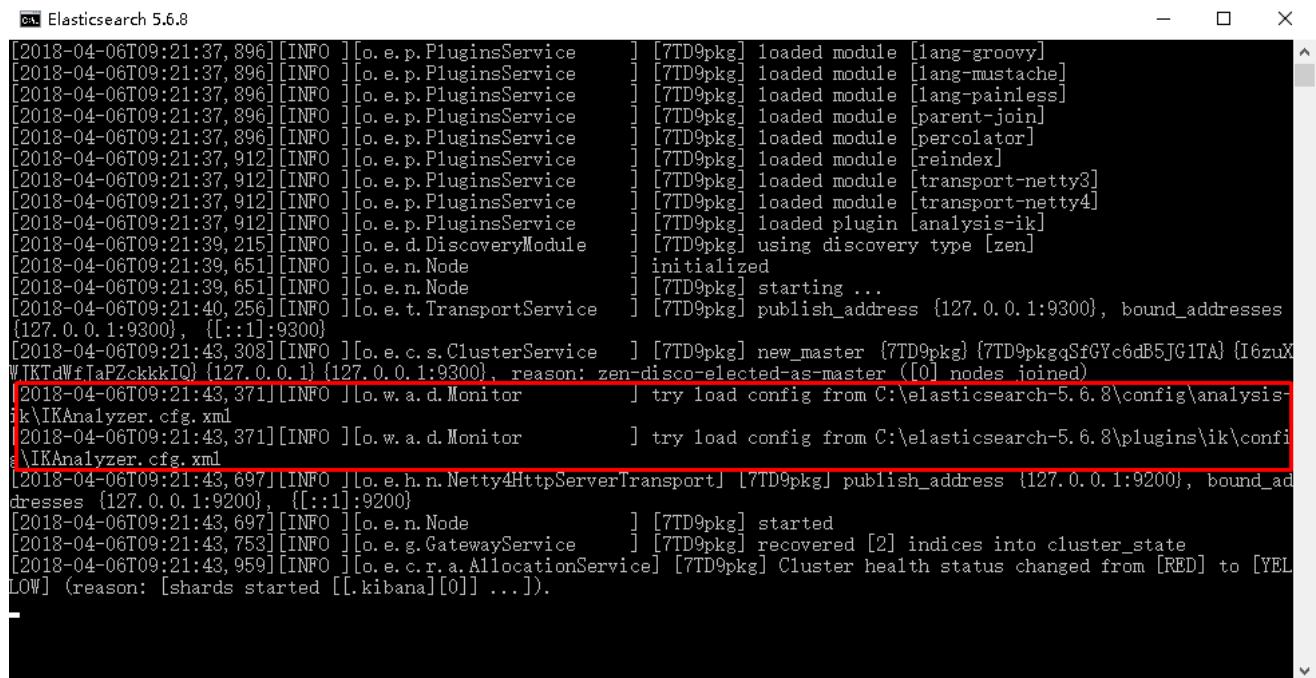


elasticsearch-a  
nalysis-ik-5.6.8.  
zip

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



```

[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              ] 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} {7TD9pkqgSfGYc6dB5JG1TA} {I6zuXWTKTdWfTaPZckkkIQ} [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        ] 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        ] 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

```
DELETE      localhost:9200/blog1
```

创建blog1索引，此时分词器使用ik\_max\_word

```
PUT      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

```
POST 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

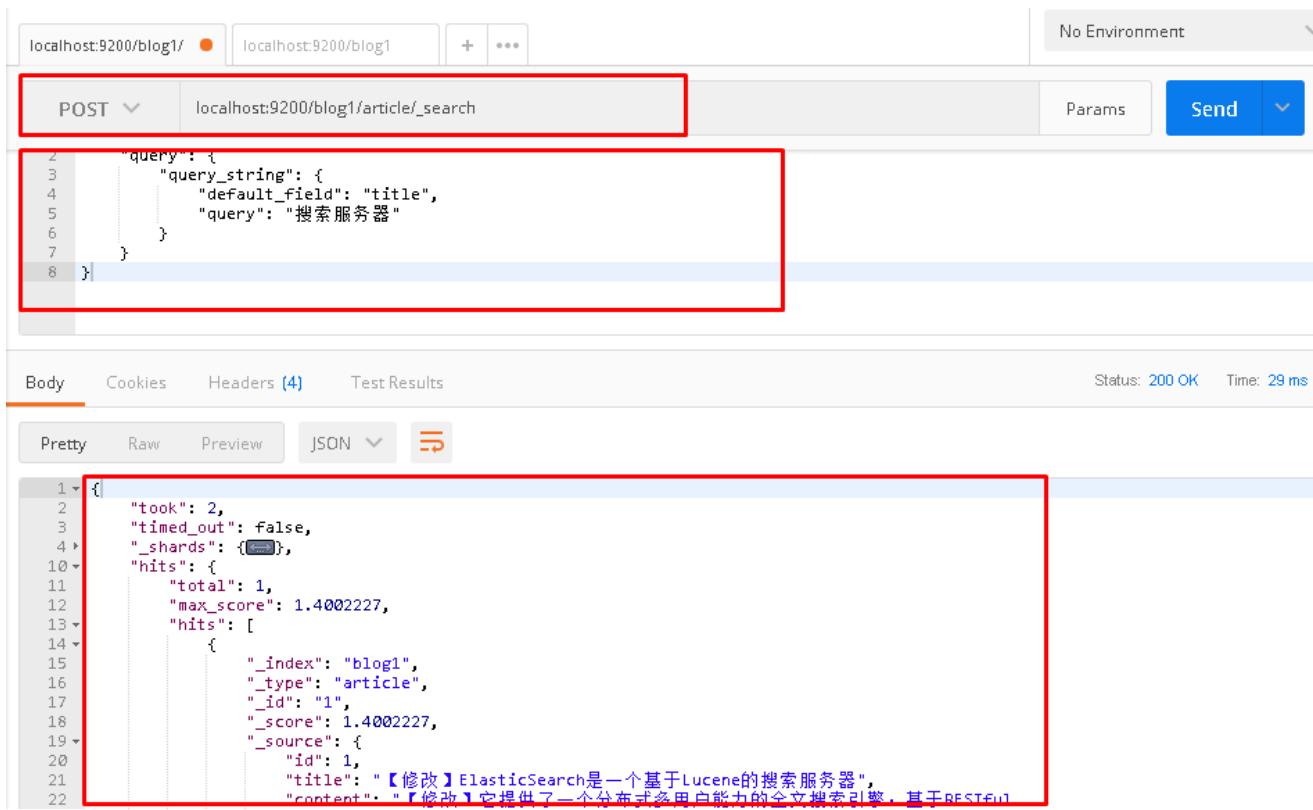
请求url：

```
POST localhost:9200/blog1/article/_search
```

请求体：

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

postman截图：



localhost:9200/blog1/ localhost:9200/blog1/article/\_search No Environment

POST localhost:9200/blog1/article/\_search Params Send

```
2 "query": {  
3     "query_string": {  
4         "default_field": "title",  
5         "query": "搜索服务器"  
6     }  
7 }  
8 }
```

Body Cookies Headers (4) Test Results Status: 200 OK Time: 29 ms

Pretty Raw Preview JSON

```
1 {  
2     "took": 2,  
3     "timed_out": false,  
4     "_shards": {  
5     },  
6     "hits": {  
7         "total": 1,  
8         "max_score": 1.4002227,  
9         "hits": [  
10             {  
11                 "_index": "blog1",  
12                 "_type": "article",  
13                 "_id": "1",  
14                 "_score": 1.4002227,  
15                 "_source": {  
16                     "id": 1,  
17                     "title": "【修改】ElasticSearch是一个基于Lucene的搜索服务器",  
18                     "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  
19                 }  
20             }  
21         }  
22     }  
23 }
```

将请求体搜索字符串修改为"钢索"，再次查询：

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

postman截图：

The screenshot shows the Postman interface with a red box highlighting the search query in the request body and another red box highlighting the search results in the response body.

**Request Body:**

```
1 {
2     "query": {
3         "query_string": {
4             "default_field": "title",
5             "query": "钢索"
6         }
7     }
8 }
```

**Response Body:**

```
1 {
2     "took": 1,
3     "timed_out": false,
4     "_shards": {
5         "total": 5,
6         "successful": 5,
7         "skipped": 0,
8         "failed": 0
9     },
10    "hits": {
11        "total": 0,
12        "max_score": null,
13        "hits": []
14    }
15 }
```

### 5.4.3 再次测试term测试

请求url :

```
POST      localhost:9200/blog1/article/_search
```

请求体 :

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

postman截图 :

The screenshot shows the Elasticsearch Dev Tools interface. At the top, there are tabs for 'localhost:9200/blog1/' and 'localhost:9200/blog1/article/\_search'. The search tab is active. Below the tabs, there is a red box highlighting the search query in the 'Body' section:

```

1 {
2   "query": {
3     "term": {
4       "title": "搜索"
5     }
6   }
7 }

```

Below the query, the status is 'Status: 200 OK' and the time taken is 'Time: 27 m'. The 'Body' tab is selected, showing the search results in a 'Pretty' JSON format. A red box highlights the first hit in the 'hits' array:

```

1 {
2   "took": 1,
3   "timed_out": false,
4   "_shards": { },
5   "hits": [
6     {
7       "total": 1,
8       "max_score": 0.25316024,
9       "hits": [
10         {
11           "_index": "blog1",
12           "_type": "article",
13           "_id": "1",
14           "_score": 0.25316024,
15           "_source": {
16             "id": 1,
17             "title": "ElasticSearch是一个基于Lucene的搜索服务器",
18             "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"
19           }
20         }
21       ]
22     }
23   ]
24 }

```

## 第六章 Elasticsearch集群

ES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，一般大于等于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一般集群中的节点数量都是3个及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。

### 6.1.3 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（意思是根本没有复制）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。

## 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: 9200  
#集群间通信端口号，在同一机器下必须不一样  
transport.tcp.port: 9300  
#设置集群自动发现机器ip集合  
discovery.zen.ping.unicast.hosts: ["127.0.0.1:9300", "127.0.0.1:9301", "127.0.0.1:9302"]
```

## 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: 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: ["127.0.0.1:9300", "127.0.0.1:9301", "127.0.0.1:9302"]
```

## 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: 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: ["127.0.0.1:9300", "127.0.0.1:9301", "127.0.0.1:9302"]
```

## 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：

```
命令提示符
Microsoft Windows [版本 10.0.16299.309]
(c) 2017 Microsoft Corporation。保留所有权利。

C:\Users\muzimoo>npm install -g grunt-cli
C:\Users\muzimoo\AppData\Roaming\npm\grunt -> C:\Users\muzimoo\AppData\Roaming\npm\node_modules\grunt-cli\bin\grunt
+ grunt-cli@1.2.0
added 16 packages in 2.578s

C:\Users\muzimoo>
```

## 启动节点2：

```
命令提示符
Microsoft Windows [版本 10.0.16299.309]
(c) 2017 Microsoft Corporation。保留所有权利。

C:\Users\muzimoo>cd C:\elasticsearch-head-master

C:\elasticsearch-head-master>grunt server
grunt-cli: The grunt command line interface (v1.2.0)

Fatal error: Unable to find local grunt.

If you're seeing this message, grunt hasn't been installed locally to
your project. For more information about installing and configuring grunt,
please see the Getting Started guide:

http://gruntjs.com/getting-started

C:\elasticsearch-head-master>
```

## 启动节点3：

```
命令提示符
Microsoft Windows [版本 10.0.16299.309]
(c) 2017 Microsoft Corporation。保留所有权利。

C:\Users\muzimoo>cd C:\elasticsearch-head-master

C:\elasticsearch-head-master>grunt server
(node:12764) ExperimentalWarning: The http2 module is an experimental API.
Running "connect:server" (connect) task
Waiting forever...
Started connect web server on http://localhost:9100
```

## 6.2.4 集群测试

### 添加索引和映射

```
PUT      localhost:9200/blog1
```

```
{
  "mappings": {
    "article": {
      "properties": {
        "id": {
          "type": "long",
          "store": true,
          "index": "not_analyzed"
        },
        "title": {
          "type": "string",
          "store": true,
          "index": "not_analyzed"
        }
      }
    }
  }
}
```

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

## 添加文档

POST      [localhost:9200/blog1/article/1](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。"  
}
```

## 使用elasticsearch-header查看集群情况

