

全文搜索引擎 Elasticsearch 入门教程

作者： 阮一峰

分享

日期： 2017年8月17日

[全文搜索](#)属于最常见的需求，开源的 [ElasticSearch](#)（以下简称 **Elastic**）是目前全文搜索引擎的首选。

它可以快速地储存、搜索和分析海量数据。维基百科、Stack Overflow、Github 都采用它。



Elastic 的底层是开源库 [Lucene](#)。但是，你没法直接用 **Lucene**，必须自己写代码去调用它的接口。**Elastic** 是 **Lucene** 的封装，提供了 **REST API** 的操作接口，开箱即用。

本文从零开始，讲解如何使用 **Elastic** 搭建自己的全文搜索引擎。每一步都有详细的说明，大家跟着做就能学会。

一、安装

Elastic 需要 **Java 8** 环境。如果你的机器还没安装 **Java**，可以参考[这篇文章](#)，注意要保证环境变量 `JAVA_HOME` 正确设置。

安装完 **Java**，就可以跟着[官方文档](#)安装 **Elastic**。直接下载压缩包比较简单。

```
$ wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-5.5.1.zip
$ unzip elasticsearch-5.5.1.zip
$ cd elasticsearch-5.5.1/
```

接着，进入解压后的目录，运行下面的命令，启动 **Elastic**。

```
$ ./bin/elasticsearch
```

如果这时[报错](#)"max virtual memory areas vm.maxmapcount [65530] is too low"，要运行下面的命令。

```
$ sudo sysctl -w vm.max_map_count=262144
```

如果一切正常，Elastic 就会在默认的9200端口运行。这时，打开另一个命令行窗口，请求该端口，会得到说明信息。

```
$ curl localhost:9200

{
  "name" : "atntrTf",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "tf9250XhQ6ee4h7YI11anA",
  "version" : {
    "number" : "5.5.1",
    "build_hash" : "19c13d0",
    "build_date" : "2017-07-18T20:44:24.823Z",
    "build_snapshot" : false,
    "lucene_version" : "6.6.0"
  },
  "tagline" : "You Know, for Search"
}
```

上面代码中，请求9200端口，Elastic 返回一个 JSON 对象，包含当前节点、集群、版本等信息。

按下 Ctrl + C，Elastic 就会停止运行。

默认情况下，Elastic 只允许本机访问，如果需要远程访问，可以修改 Elastic 安装目录的 config/elasticsearch.yml 文件，去掉 network.host 的注释，将它的值改成 0.0.0.0，然后重新启动 Elastic。

```
network.host: 0.0.0.0
```

上面代码中，设成 0.0.0.0 让任何人都可以访问。线上服务不要这样设置，要设成具体的 IP。

二、基本概念

2.1 Node 与 Cluster

Elastic 本质上是一个分布式数据库，允许多台服务器协同工作，每台服务器可以运行多个 Elastic 实例。

单个 Elastic 实例称为一个节点（node）。一组节点构成一个集群（cluster）。

2.2 Index

Elastic 会索引所有字段，经过处理后写入一个反向索引（Inverted Index）。查找数据的时候，直接查找该索引。

所以，Elastic 数据管理的顶层单位就叫做 Index（索引）。它是单个数据库的同义词。每个 Index（即数据库）的名字必须是小写。

下面的命令可以查看当前节点的所有 Index。

```
$ curl -X GET 'http://localhost:9200/_cat/indices?v'
```

2.3 Document

Index 里面单条的记录称为 Document（文档）。许多条 Document 构成了一个 Index。

Document 使用 JSON 格式表示，下面是一个例子。

```
{
  "user": "张三",
  "title": "工程师",
  "desc": "数据库管理"
}
```

同一个 Index 里面的 Document，不要求有相同的结构（scheme），但是最好保持相同，这样有利于提高搜索效率。

2.4 Type

Document 可以分组，比如 `weather` 这个 Index 里面，可以按城市分组（北京和上海），也可以按气候分组（晴天和雨天）。这种分组就叫做 Type，它是虚拟的逻辑分组，用来过滤 Document。

不同的 Type 应该有相似的结构（schema），举例来说，`id` 字段不能在这个组是字符串，在另一个组是数值。这是与关系型数据库的表的[一个区别](#)。性质完全不同的数据（比如 `products` 和 `logs`）应该存成两个 Index，而不是一个 Index 里面的两个 Type（虽然可以做到）。

下面的命令可以列出每个 Index 所包含的 Type。

```
$ curl 'localhost:9200/_mapping?pretty=true'
```

根据[规划](#)，Elastic 6.x 版只允许每个 Index 包含一个 Type，7.x 版将会彻底移除 Type。

三、新建和删除 Index

新建 Index，可以直接向 Elastic 服务器发出 PUT 请求。下面的例子是新建一个名叫 `weather` 的 Index。

```
$ curl -X PUT 'localhost:9200/weather'
```

服务器返回一个 JSON 对象，里面的 `acknowledged` 字段表示操作成功。

```
{
  "acknowledged":true,
  "shards_acknowledged":true
}
```

然后，我们发出 **DELETE** 请求，删除这个 **Index**。

```
$ curl -X DELETE 'localhost:9200/weather'
```

四、中文分词设置

首先，安装中文分词插件。这里使用的是 [ik](#)，也可以考虑其他插件（比如 [smarten](#)）。

```
$ ./bin/elasticsearch-plugin install https://github.com/medcl/elasticsearch-plugins/tree/master/analysis-ik
```

上面代码安装的是5.5.1版的插件，与 **Elastic 5.5.1** 配合使用。

接着，重新启动 **Elastic**，就会自动加载这个新安装的插件。

然后，新建一个 **Index**，指定需要分词的字段。这一步根据数据结构而异，下面的命令只针对本文。基本上，凡是需要搜索的中文字段，都要单独设置一下。

```
$ curl -X PUT 'localhost:9200/accounts' -d '{
  "mappings": {
    "person": {
      "properties": {
        "user": {
          "type": "text",
          "analyzer": "ik_max_word",
          "search_analyzer": "ik_max_word"
        },
        "title": {
          "type": "text",
          "analyzer": "ik_max_word",
          "search_analyzer": "ik_max_word"
        },
        "desc": {
          "type": "text",
          "analyzer": "ik_max_word",
          "search_analyzer": "ik_max_word"
        }
      }
    }
  }
}
```

上面代码中，首先新建一个名称为 **accounts** 的 **Index**，里面有一个名称为 **person** 的 **Type**。**person** 有三个字段。

- user
- title
- desc

这三个字段都是中文，而且类型都是文本（text），所以需要指定中文分词器，不能使用默认的英文分词器。

Elastic 的分词器称为 [analyzer](#)。我们对每个字段指定分词器。

```
"user": {
  "type": "text",
  "analyzer": "ik_max_word",
  "search_analyzer": "ik_max_word"
}
```

上面代码中，`analyzer` 是字段文本的分词器，`search_analyzer` 是搜索词的分词器。`ik_max_word` 分词器是插件 `ik` 提供的，可以对文本进行最大数量的分词。

五、数据操作

5.1 新增记录

向指定的 `/Index/Type` 发送 `PUT` 请求，就可以在 `Index` 里面新增一条记录。比如，向 `/accounts/person` 发送请求，就可以新增一条人员记录。

```
$ curl -X PUT 'localhost:9200/accounts/person/1' -d '{
  "user": "张三",
  "title": "工程师",
  "desc": "数据库管理"
}'
```

服务器返回的 `JSON` 对象，会给出 `Index`、`Type`、`Id`、`Version` 等信息。

```
{
  "_index": "accounts",
  "_type": "person",
  "_id": "1",
  "_version": 1,
  "result": "created",
  "_shards": {"total": 2, "successful": 1, "failed": 0},
  "created": true
}
```

如果你仔细看，会发现请求路径是 `/accounts/person/1`，最后的 `1` 是该条记录的 `Id`。它不一定是数字，任意字符串（比如 `abc`）都可以。

新增记录的时候，也可以不指定 `Id`，这时要改成 `POST` 请求。

```
$ curl -X POST 'localhost:9200/accounts/person' -d '{
  "user": "李四",
  "title": "工程师",
  "desc": "系统管理"
}'
```

上面代码中，向 `/accounts/person` 发出一个 **POST** 请求，添加一个记录。这时，服务器返回的 **JSON** 对象里面，`_id` 字段就是一个随机字符串。

```
{
  "_index": "accounts",
  "_type": "person",
  "_id": "AV3qGfrC6jMbsbXb6k1p",
  "_version": 1,
  "result": "created",
  "_shards": {"total": 2, "successful": 1, "failed": 0},
  "created": true
}
```

注意，如果没有先创建 **Index**（这个例子是 `accounts`），直接执行上面的命令，**Elastic** 也不会报错，而是直接生成指定的 **Index**。所以，打字的时候要小心，不要写错 **Index** 的名称。

5.2 查看记录

向 `/Index/Type/Id` 发出 **GET** 请求，就可以查看这条记录。

```
$ curl 'localhost:9200/accounts/person/1?pretty=true'
```

上面代码请求查看 `/accounts/person/1` 这条记录，URL 的参数 `pretty=true` 表示以易读的格式返回。

返回的数据中，`found` 字段表示查询成功，`_source` 字段返回原始记录。

```
{
  "_index" : "accounts",
  "_type" : "person",
  "_id" : "1",
  "_version" : 1,
  "found" : true,
  "_source" : {
    "user" : "张三",
    "title" : "工程师",
    "desc" : "数据库管理"
  }
}
```

如果 **Id** 不正确，就查不到数据，`found` 字段就是 `false`。

```
$ curl 'localhost:9200/weather/beijing/abc?pretty=true'

{
  "_index" : "accounts",
  "_type" : "person",
  "_id" : "abc",
  "found" : false
}
```

5.3 删除记录

删除记录就是发出 **DELETE** 请求。

```
$ curl -X DELETE 'localhost:9200/accounts/person/1'
```

这里先不要删除这条记录，后面还要用到。

5.4 更新记录

更新记录就是使用 **PUT** 请求，重新发送一次数据。

```
$ curl -X PUT 'localhost:9200/accounts/person/1' -d '
{
  "user" : "张三",
  "title" : "工程师",
  "desc" : "数据库管理，软件开发"
}'

{
  "_index":"accounts",
  "_type":"person",
  "_id":"1",
  "_version":2,
  "result":"updated",
  "_shards":{"total":2,"successful":1,"failed":0},
  "created":false
}
```

上面代码中，我们将原始数据从"数据库管理"改成"数据库管理，软件开发"。返回结果里面，有几个字段发生了变化。

```
"_version" : 2,
"result" : "updated",
"created" : false
```

可以看到，记录的 **Id** 没变，但是版本（**version**）从 **1** 变成 **2**，操作类型（**result**）从 **created** 变成 **updated**，**created** 字段变成 **false**，因为这次不是新建记录。

六、数据查询

6.1 返回所有记录

使用 **GET** 方法，直接请求 `/Index/Type/_search`，就会返回所有记录。

```
$ curl 'localhost:9200/accounts/person/_search'

{
  "took":2,
  "timed_out":false,
  "_shards":{"total":5,"successful":5,"failed":0},
  "hits":{"
    "total":2,
    "max_score":1.0,
    "hits":[
      {
        "_index":"accounts",
        "_type":"person",
        "_id":"AV3qGfrC6jMbsbXb6k1p",
        "_score":1.0,
        "_source": {
          "user": "李四",
          "title": "工程师",
          "desc": "系统管理"
        }
      },
      {
        "_index":"accounts",
        "_type":"person",
        "_id":"1",
        "_score":1.0,
        "_source": {
          "user": "张三",
          "title": "工程师",
          "desc": "数据库管理, 软件开发"
        }
      }
    ]
  }
}
```

上面代码中，返回结果的 **took** 字段表示该操作的耗时（单位为毫秒），**timed_out** 字段表示是否超时，**hits** 字段表示命中的记录，里面子字段的含义如下。

- **total**：返回记录数，本例是2条。
- **max_score**：最高的匹配程度，本例是1.0。
- **hits**：返回的记录组成的数组。

返回的记录中，每条记录都有一个 **_score** 字段，表示匹配的程序，默认是按照这个字段降序排列。

6.2 全文搜索

Elastic 的查询非常特别，使用自己的[查询语法](#)，要求 **GET** 请求带有数据体。

```
$ curl 'localhost:9200/accounts/person/_search' -d '{
  "query" : { "match" : { "desc" : "软件" }}
}'
```


上面代码使用 [Match 查询](#)，指定的匹配条件是 `desc` 字段里面包含"软件"这个词。返回结果如下。

```
{
  "took":3,
  "timed_out":false,
  "_shards":{"total":5,"successful":5,"failed":0},
  "hits":{"
    "total":1,
    "max_score":0.28582606,
    "hits":[
      {
        "_index":"accounts",
        "_type":"person",
        "_id":"1",
        "_score":0.28582606,
        "_source": {
          "user" : "张三",
          "title" : "工程师",
          "desc" : "数据库管理，软件开发"
        }
      }
    ]
  }
}
```

Elastic 默认一次返回10条结果，可以通过 `size` 字段改变这个设置。

```
$ curl 'localhost:9200/accounts/person/_search' -d '
{
  "query" : { "match" : { "desc" : "管理" }},
  "size": 1
}'
```

上面代码指定，每次只返回一条结果。

还可以通过 `from` 字段，指定位移。

```
$ curl 'localhost:9200/accounts/person/_search' -d '
{
  "query" : { "match" : { "desc" : "管理" }},
  "from": 1,
  "size": 1
}'
```

上面代码指定，从位置1开始（默认是从位置0开始），只返回一条结果。

6.3 逻辑运算

如果有多个搜索关键字，Elastic 认为它们是 `or` 关系。

```
$ curl 'localhost:9200/accounts/person/_search' -d '
```

```
{
  "query" : { "match" : { "desc" : "软件 系统" }}
}'
```

上面代码搜索的是 软件 or 系统 。

如果要执行多个关键词的 and 搜索，必须使用[布尔查询](#)。




```
$ curl 'localhost:9200/accounts/person/_search' -d '{
  "query": {
    "bool": {
      "must": [
        { "match": { "desc": "软件" } },
        { "match": { "desc": "系统" } }
      ]
    }
  }
}'
```

七、参考链接

- [ElasticSearch 官方手册](#)
- [A Practical Introduction to Elasticsearch](#)

（完）

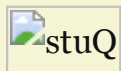
文档信息

- 版权声明： 自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](#)）
- 发表日期： 2017年8月17日
- 更多内容： [档案](#) » [开发者手册](#)
- 博客文集： 《前方的路》， 《未来世界的幸存者》
- 社交媒体：  twitter,  weibo
- Feed订阅： 

打造中国最权威的《前端-全栈-工程化课程》

八年专注前端， 珠峰培训让你高薪就业

快戳我！了解详情 



相关文章

- 2017.07.29: [窗口管理器 xmonad 教程](#)

开发者最需要的，就是一个顺手的开发环境。

▪ 2017.07.18: [Pull Request 的命令行管理](#)

Github 的一大特色就是 Pull Request 功能（简写为 PR）。

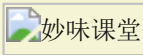
▪ 2017.06.22: [HTML 自定义元素教程](#)

组件是 Web 开发的方向，现在的热点是 JavaScript 组件，但是 HTML 组件未来可能更有希望。

▪ 2017.06.15: [树莓派新手入门教程](#)

树莓派（Raspberry Pi）是学习计算机知识、架设服务器的好工具，价格低廉，可玩性高。

广告（购买广告位）



shadowsocks科学上网

免费体验 不限流量



稳定 快速 实惠


倚天剑 shadow

