# Elasticsearch

## Elasticsearch概述

### 选择检索引擎

随着央视诗词大会的热播，小史开始对诗词感兴趣，最喜欢的就是飞花令的环节。老师正好也来了兴趣，想考考小史。

*老师：请说出带“前”字的诗句。*

*小史：啊？忘了。*

*老师：小史，那你背一下李白的静夜思。*

*小史：那简单，床前明月光，疑是地上霜…等等，这句就有前啊，哎呀！*

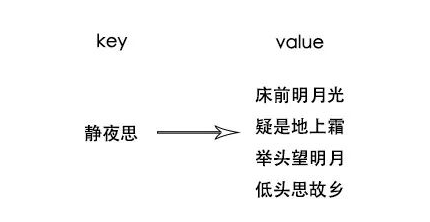
*老师：你再背一下李白的望卢川瀑布。*

*小史：日照香炉生紫烟，遥看瀑布挂前川…哎呀！*

*……*

我们可以发现，带有前的诗句其实很多，很多都是我们耳熟能详的，但是我们为什么会想不起来呢？

我们平时背书的时候，都是从前往后，先记诗名、作者，然后背诗的内容，那么在我们的脑袋里，可能就是这么建立的索引：

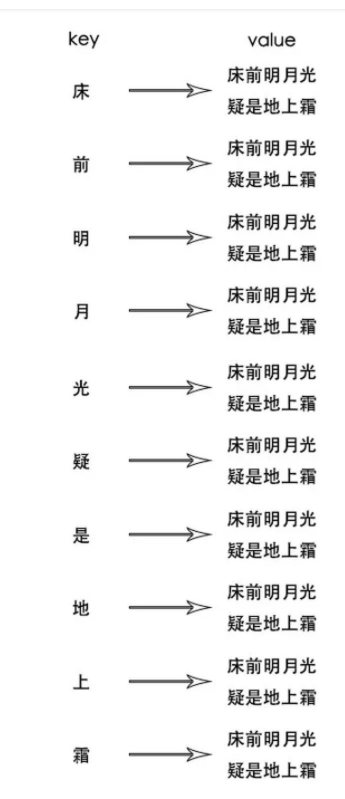


以静夜思作为key，内容作为value。一说到静夜思，我们就能立刻想起它里面的诗句，这中索引方式也被称为正排索引。

一旦当老师要求说出带“前”字的诗句时，由于没有索引，就只能遍历脑海中的所有诗句，当脑海中诗词量大的时候，就很难在短时间内得到结果了。

通过上面的例子，也就引发出一个思考，我们是否可以通过以“前”作为索引的key，诗的内容作为value呢？

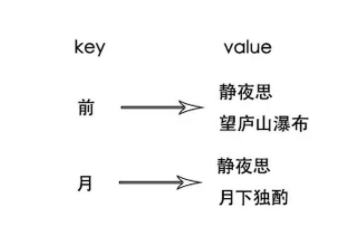
当然可以，我们可以将诗句中的一个个词取出来，作为索引的key，value对应一句诗：



这样当我们想找带有“前”的诗，我们很快就能找到对应的诗句。这也是今天的主角--------倒排索引（也叫反向索引）。

当然，如果实际中，我们真要去这么背诵的话，记忆量会成倍的增加。在计算机中提现出来则是：为了提高检索效率，会占用大量的缓存。

我们可以再优化一下，还是以诗中的一个个词作为key，value则是换为诗名。



然后我们获取到这些诗之后，再正排索引（正向索引）去检索，就能很快地得到我们想要的结果。

基于倒排索引这种检索方式，业界很早就有了一个叫做lecene的库，用它可以很方便的地建立倒排索引。

但是 Lucene 还是一个库，必须要懂一点搜索引擎原理的人才能用的好，所以后来又有人基于 Lucene 进行封装，写出了 Elasticsearch。

### 全文搜索引擎

Google，百度类的网站搜索，它们都是根据网页中的关键字生成索引，我们在搜索的时候输入关键字，它们会将该关键字即索引匹配到的所有网页返回；还有常见的项目中应用日志的搜索等等。对于这些非结构化的数据文本，关系型数据库搜索不是能很好的支持。

一般传统数据库，全文检索都实现的很鸡肋，因为一般也没人用数据库存文本字段。进行全文检索需要扫描整个表，如果数据量大的话即使对 SQL 的语法优化，也收效甚微。建立了索引，但是维护起来也很麻烦，对于 insert 和 update 操作都会重新构建索引。

基于以上原因可以分析得出，在一些生产环境中，使用常规的搜索方式，性能是非常差的：

* 搜索的数据对象是大量的非结构化的文本数据。
* 文件记录量达到数十万或数百万个甚至更多。
* 支持大量基于交互式文本的查询。
* 需求非常灵活的全文搜索查询。
* 对高度相关的搜索结果的有特殊需求，但是没有可用的关系数据库可以满足。
* 对不同记录类型、非文本数据操作或安全事务处理的需求相对较少的情况。

为了解决结构化数据搜索和非结构化数据搜索性能问题，我们就需要专业，健壮，强大的全文搜索引擎。

这里说到的全文搜索引擎指的是目前广泛应用的主流搜索引擎。它的工作原理是计算机索引程序通过扫描文章中的每一个词，对每一个词建立一个索引，指明该词在文章中出现的次数和位置，当用户查询时，检索程序就根据事先建立的索引进行查找，并将查找的结果反馈给用户的检索方式。这个过程类似于通过字典中的检索字表查字的过程。

### Elastic Stack

The Elastic Stack, 包括 Elasticsearch、Kibana、Beats 和 Logstash（也称为 ELK Stack）。能够安全可靠地获取任何来源、任何格式的数据，然后实时地对数据进行搜索、分析和可视化。Elaticsearch，简称为 ES，ES 是一个开源的高扩展的分布式全文搜索引擎，是整个 ElasticStack 技术栈的核心。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。

### Elasticsearch 应用案例

* GitHub: 2013 年初，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”。
* 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 Elasticsearch 分析处理 32 亿条实时日志。
* 阿里：使用 Elasticsearch 构建日志采集和分析体系。
* Stack Overflow：解决 Bug 问题的网站，全英文，编程人员交流的网站。

## Elasticsearch 入门

### Elasticsearch 安装

这里推荐docker安装elasticsearch

|  |
| --- |
| docker run --name elasticsearch -d \  -v /data/elasticsearch/data:/usr/share/elasticsearch/data \  -v /data/elasticsearch/logs:/usr/share/elasticsearch/logs \  -p 9203:9200 \  -p 9303:9300 \  -e ES\_JAVA\_OPTS="-Xms256m -Xmx256m" \  -e "discovery.type=single-node" \  elasticsearch:7.8.0 |

注意：9300 端口为 Elasticsearch 集群间组件的通信端口，9200 端口为浏览器访问的 http协议 RESTful 端口。

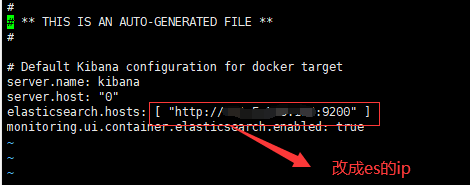
### Kibana安装

这里推荐docker安装kibana

|  |
| --- |
| docker run --restart=always -d -e "elasticsearch.hosts==http://es-ip:9200" -p 5601:5601 --name kibana kibana:7.8.0 |

如果是防火墙已经暴露端口的情况下，这么配置之后直接就可以正常连接，如果不行的话，需要进入kibana容器显示配置一下elasticsearch的ip。

|  |
| --- |
| docker exec -it kibana /bin/bash  cd /usr/share/kibana/config/  vi kibana.yml |



保存后重启docker容器。

### 核心概念

#### 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），并且当我们要对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。 能搜索的数据必须索引，这样的好处是可以提高查询速度，比如：新华字典前面的目录 就是索引的意思，目录可以提高查询速度。

#### 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。不同的版本，类型发生了不同的变化

|  |  |
| --- | --- |
| 版本 | Type |
| 5.x | 支持多种 type |
| 6.x | 只能有一种 type |
| 7.x | 默认不再支持自定义索引类型（默认类型为：\_doc） |
| 8.x | 将要去掉type |

#### 文档（Document）

一个文档是一个可被索引的基础信息单元，也就是一条数据 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。

#### 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。

#### 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。

#### 分片（Shards）

一个索引可以存储超出单个节点硬件限制的大量数据。比如，一个具有 10 亿文档数据 的索引占据 1TB 的磁盘空间，而任一节点都可能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， 每一份就称之为分片。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，它的文档怎样聚合和搜索请求，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，无需过分关心。

#### 副本（Replicas）

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(副本)。

复制分片之所以重要，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副本上并行运行。

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 1 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 1 个主分片和另外 1 个复制分片（1 个完全拷贝），这样的话每个索引总共就有 2 个分片，我们需要根据索引需要确定分片个数。

### Elasticsearch 基本操作

#### RESTful

REST 指的是一组架构约束条件和原则。满足这些约束条件和原则的应用程序或设计就 是 RESTful。Web 应用程序最重要的 REST 原则是，客户端和服务器之间的交互在请求之 间是无状态的。从客户端到服务器的每个请求都必须包含理解请求所必需的信息。如果服务 器在请求之间的任何时间点重启，客户端不会得到通知。此外，无状态请求可以由任何可用 服务器回答，这十分适合云计算之类的环境。客户端可以缓存数据以改进性能。

在服务器端，应用程序状态和功能可以分为各种资源。资源是一个有趣的概念实体，它 向客户端公开。资源的例子有：应用程序对象、数据库记录、算法等等。每个资源都使用 URI (Universal Resource Identifier) 得到一个唯一的地址。所有资源都共享统一的接口，以便在客 户端和服务器之间传输状态。使用的是标准的 HTTP 方法，比如 GET、PUT、POST 和 DELETE。

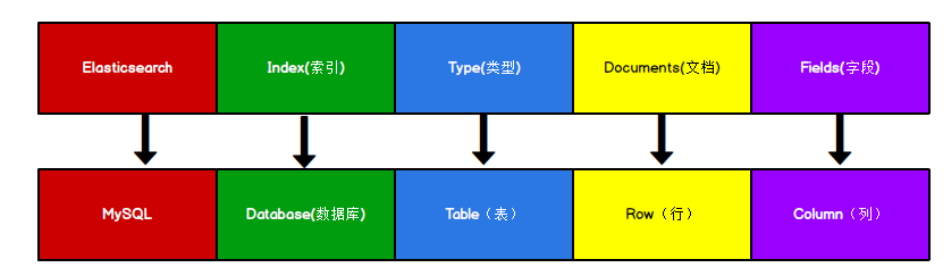
在 REST 样式的 Web 服务中，每个资源都有一个地址。资源本身都是方法调用的目 标，方法列表对所有资源都是一样的。这些方法都是标准方法，包括 HTTP GET、POST、 PUT、DELETE，还可能包括 HEAD 和 OPTIONS。简单的理解就是，如果想要访问互联 网上的资源，就必须向资源所在的服务器发出请求，请求体中必须包含资源的网络路径，以 及对资源进行的操作(增删改查)。

#### 客户端请求

如果直接通过浏览器向 Elasticsearch 服务器发请求，那么需要在发送的请求中包含 HTTP 标准的方法，而 HTTP 的大部分特性且仅支持 GET 和 POST 方法。所以为了能方便 地进行客户端的访问，可以使用 Postman 软件 Postman 是一款强大的网页调试工具，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。

#### 数据格式

Elasticsearch 是面向文档型数据库，一条数据在这里就是一个文档。为了方便大家理解， 我们将 Elasticsearch 里存储文档数据和关系型数据库 MySQL 存储数据的概念进行一个类比 ES 里的 Index 可以看做一个库，而 Types 相当于表，Documents 则相当于表的行。 这里 Types 的概念已经被逐渐弱化，Elasticsearch 6.X 中，一个 index 下已经只能包含一个 type，Elasticsearch 7.X 中, Type 的概念已经被删除了。



用 JSON 作为文档序列化的格式，比如一条用户信息：



### HTTP 操作

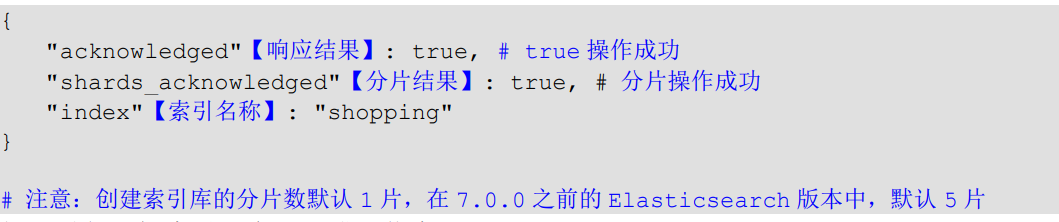
#### 索引操作

##### 创建索引

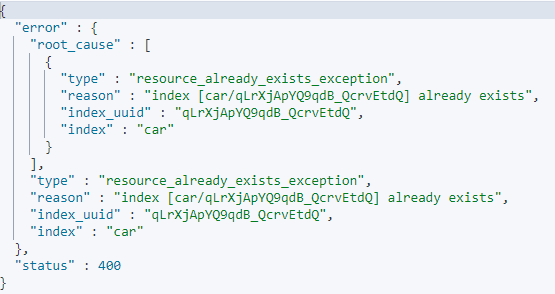
让我们简单地创建一个car索引

|  |
| --- |
| PUT /car |

就会返回创建成功的消息：



如果重复添加索引，会返回错误信息

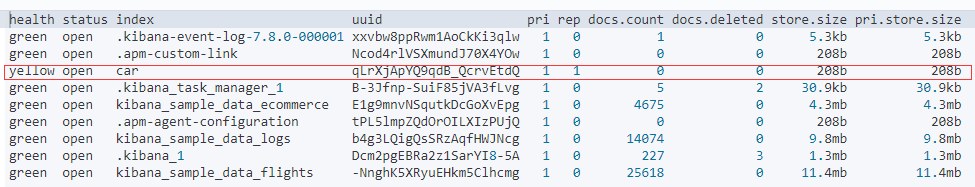


##### 查看所有索引

查看所有索引

|  |
| --- |
| GET /\_cat/indices?v |

结果：

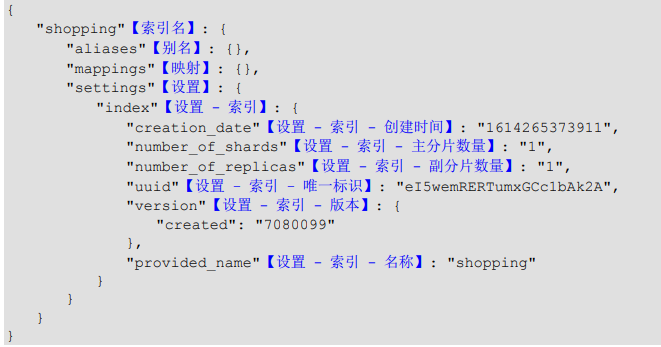




##### 查看单个索引

|  |
| --- |
| GET /car |

结果

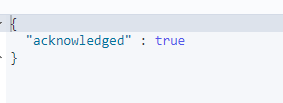


##### 删除索引

删除car

|  |
| --- |
| DELETE car |

返回结果



#### 文档操作

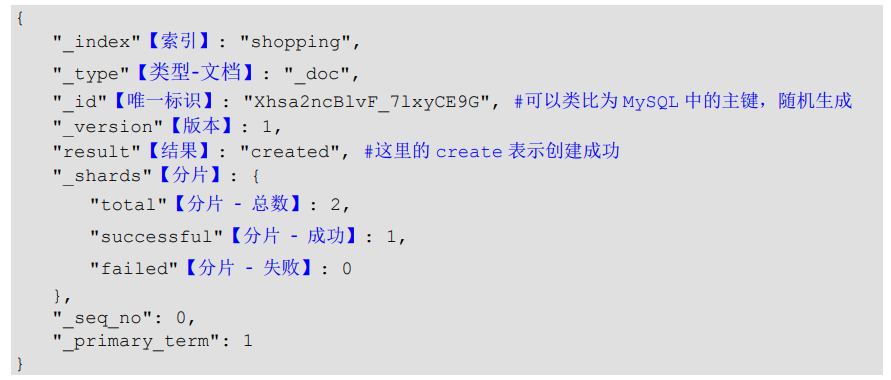
##### 创建文档

上面已经讲了创建索引，接下来我们来创建文档，并添加数据。这里的文档可以类比为关系型数据库中的表数据，添加的数据格式为 JSON 格式。

我们也可以不直接创建索引，而是可以文档和索引一起创建：

|  |
| --- |
| POST /shopping/\_doc  {  "title":"小米手机",  "category":"小米",  "images":"http://www.gulixueyuan.com/xm.jpg",  "price":3999.00  } |

服务器响应结果：



上面的数据创建后，由于没有指定数据唯一性标识（ID），默认情况下，ES 服务器会随机 生成一个。

如果想要自定义唯一性标识，需要在创建时指定



服务器响应结果：



此处需要注意：如果增加数据时明确数据主键，那么请求方式也可以为 PUT

##### 查看文档

查看文档时，需要指明文档的唯一性标识，类似于 MySQL 中数据的主键查询。

|  |
| --- |
| GET /shopping/\_doc/1 |

查询成功后，服务器响应结果：

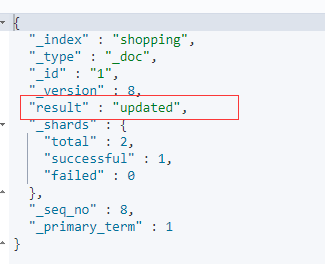


##### 修改文档

和新增文档一样，输入相同的 URL 地址请求，如果请求体变化，会将原有的数据内容覆盖。不同的是，这里必须指定id。

|  |
| --- |
| POST /shopping/\_doc/1  {  "title":"华为手机",  "category":"华为",  "images":"http://www.gulixueyuan.com/hw.jpg",  "price":4999.00  } |

服务器返回结果



##### 修改字段

修改数据时，也可以只修改某一给条数据的局部信息。

|  |
| --- |
| POST /shopping/\_update/1  {  "doc": {  "price":3000.00  }  } |

服务器返回结果



##### 删除文档

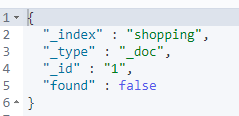
删除一个文档不会立即从磁盘上移除，它只是被标记成已删除（逻辑删除）。

|  |
| --- |
| DELETE /shopping/\_doc/1 |

服务器返回结果



删除后再查询当前文档信息



如果删除一个并不存在的文档



##### 条件删除文档

一般删除数据都是根据文档的唯一性标识进行删除，实际操作时，也可以根据条件对多条数 据进行删除。

首先分别增加多条数据:

|  |
| --- |
| {  "title":"小米手机",  "category":"小米",  "images":"http://www.gulixueyuan.com/xm.jpg",  "price":4000.00  }  {  "title":"华为手机",  "category":"华为",  "images":"http://www.gulixueyuan.com/hw.jpg",  "price":4000.00  } |

向 ES 服务器发 POST 请求 ：/shopping/\_delete\_by\_query

请求体内容为：

|  |
| --- |
| {  "query":{  "match":{  "price":4000.00  }  }  } |

删除成功后，服务器响应结果：

