一、安装部署

环境

# 环境：Ubuntu-20.04

# 用户：developer

# 下载 Java 8

# https://www.java.com/zh\_CN/download/linux\_manual.jsp

#https://sdlc-esd.oracle.com/ESD6/JSCDL/jdk/8u181-b13/96a7b8442fe848ef90c96a2fad6ed6d1/jre-8u181-linux-x64.tar.gz?GroupName=JSC&FilePath=/ESD6/JSCDL/jdk/8u181-b13/96a7b8442fe848ef90c96a2fad6ed6d1/jre-8u181-linux-x64.tar.gz&BHost=javadl.sun.com&File=jre-8u181-linux-x64.tar.gz&AuthParam=1539426463\_6e8176907d5960b39ae96052012bc3c5&ext=.gz

# 安装 Java 8（JDK 1.8）

tar -zxvf jre-8u181-linux-x64.tar.gz

mv jre1.8.0\_181 /usr/local/jre1.8.0\_181

# 添加到环境变量

vi /etc/profile

export JAVA\_HOME=/usr/local/jre1.8.0\_181

export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

export PATH=$PATH:$JAVA\_HOME/bin

# 使全局变量即时生效

source /etc/profile

# 查看 Java 版本信息

java -version

java version "1.8.0\_181"

Java(TM) SE Runtime Environment (build 1.8.0\_181-b13)

Java HotSpot(TM) 64-Bit Server VM (build 25.181-b13, mixed mode)

安装部署

# 下载 elasticsearch-6.7.2.tar.gz

# https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.7.2.tar.gz

# 安装 elasticsearch-6.7.2

tar -zxvf elasticsearch-6.7.2.tar.gz

mv elasticsearch-6.7.2 /usr/local/elasticsearch-6.7.2

mkdir /usr/local/elasticsearch-6.7.2/var

mkdir /usr/local/elasticsearch-6.7.2/data

mkdir /usr/local/elasticsearch-6.7.2/logs

chown -R developer:developer /usr/local/elasticsearch-6.7.2

# 下载 elasticsearch-analysis-ik-6.7.2.zip

#https://github.com/medcl/elasticsearch-analysis-ik/releases/download/v6.7.2/elasticsearch-analysis-ik-6.7.2.zip

# 安装 elasticsearch-analysis-ik-6.7.2

/usr/local/elasticsearch-6.7.2/bin/elasticsearch-plugin install https://github.com/medcl/elasticsearch-analysis-ik/releases/download/v6.7.2/elasticsearch-analysis-ik-6.7.2.zip

# 修改配置文件

vi /usr/local/elasticsearch-6.7.2/config/elasticsearch.yml

cluster.name: es-wsl

network.host: 127.0.0.1

path.data: /usr/local/elasticsearch-6.7.2/data

path.logs: /usr/local/elasticsearch-6.7.2/logs

http.port: 9200

http.cors.enabled: true

http.cors.allow-origin: "\*"

http.cors.allow-methods: OPTIONS,HEAD,GET,POST,PUT,PATCH,DELETE

http.cors.allow-headers: Authorization

# 解决错误：分配的最大虚拟内存不足

# max virtual memory areas vm.max\_map\_count [65530] is too low, increase to at least [262144]

vi /etc/sysctl.conf

添加：

vm.max\_map\_count=262144

# 使 sysctl.conf 立即生效

sysctl -p

# 解决错误：限制的最大文件句柄数太小

# max file descriptors [4096] for elasticsearch process is too low, increase to at least [65535]

vi /etc/security/limits.conf

添加：

developer soft nofile 65535

developer hard nofile 65535

developer memlock unlimited

# 使 limits.conf 立即生效

su developer

# 查看最大文件句柄数

ulimit -Hn

# 以 developer 用户身份启动服务，elasticsearch 不支持以 root 用户身份启动

# -d，后台运行

# -p，指定 pid 文件路径

/usr/local/elasticsearch-6.7.2/bin/elasticsearch

/usr/local/elasticsearch-6.7.2/bin/elasticsearch -d -p /usr/local/elasticsearch-6.7.2/var/run.pid

# 以 root 用户身份启动服务

su - developer -c "/usr/local/elasticsearch-6.7.2/bin/elasticsearch"

su - developer -c "/usr/local/elasticsearch-6.7.2/bin/elasticsearch -d -p /usr/local/elasticsearch-6.7.2/var/run.pid"

# 停止服务

kill `cat /usr/local/elasticsearch-6.7.2/var/run.pid`

# 查看版本号

curl http://127.0.0.1:9200

# 验证中文分词

curl -X POST 'http://127.0.0.1:9200/\_analyze?pretty=true' \

-H 'Content-Type: application/json' \

-d '{"text":"中华人民共和国CHINESE","tokenizer":"ik\_max\_word"}'

{

"tokens": [

{

"token": "中华人民共和国",

"start\_offset": 0,

"end\_offset": 7,

"type": "CN\_WORD",

"position": 0

},

{

"token": "中华人民",

"start\_offset": 0,

"end\_offset": 4,

"type": "CN\_WORD",

"position": 1

},

{

"token": "中华",

"start\_offset": 0,

"end\_offset": 2,

"type": "CN\_WORD",

"position": 2

},

{

"token": "华人",

"start\_offset": 1,

"end\_offset": 3,

"type": "CN\_WORD",

"position": 3

},

{

"token": "人民共和国",

"start\_offset": 2,

"end\_offset": 7,

"type": "CN\_WORD",

"position": 4

},

{

"token": "人民",

"start\_offset": 2,

"end\_offset": 4,

"type": "CN\_WORD",

"position": 5

},

{

"token": "共和国",

"start\_offset": 4,

"end\_offset": 7,

"type": "CN\_WORD",

"position": 6

},

{

"token": "共和",

"start\_offset": 4,

"end\_offset": 6,

"type": "CN\_WORD",

"position": 7

},

{

"token": "国",

"start\_offset": 6,

"end\_offset": 7,

"type": "CN\_CHAR",

"position": 8

},

{

"token": "chinese",

"start\_offset": 7,

"end\_offset": 14,

"type": "ENGLISH",

"position": 9

}

]

}

二、Hello World

创建文档

curl -X PUT 'http://127.0.0.1:9200/documents/\_doc/1' \

--header 'Content-Type: application/json' \

--data-raw '{

"id":1,

"content":"Hello World"

}'

{

"\_index": "documents",

"\_type": "\_doc",

"\_id": "1",

"\_version": 1,

"result": "created",

"\_shards": {

"total": 2,

"successful": 1,

"failed": 0

},

"\_seq\_no": 0,

"\_primary\_term": 1

}

查询文档

curl -X GET 'http://127.0.0.1:9200/documents/\_doc/1'

{

"\_index": "documents",

"\_type": "\_doc",

"\_id": "1",

"\_version": 1,

"\_seq\_no": 0,

"\_primary\_term": 1,

"found": true,

"\_source": {

"id": 1,

"content": "Hello World"

}

}

搜索文档

curl -X GET 'http://127.0.0.1:9200/documents/\_search?q=content:hello'

{

"took": 1,

"timed\_out": false,

"\_shards": {

"total": 5,

"successful": 5,

"skipped": 0,

"failed": 0

},

"hits": {

"total": 1,

"max\_score": 0.2876821,

"hits": [

{

"\_index": "documents",

"\_type": "\_doc",

"\_id": "1",

"\_score": 0.2876821,

"\_source": {

"id": 1,

"content": "Hello World"

}

}

]

}

}

三、基本概念

ElasticSearch

是一个基于 Lucene 的分布式、高扩展、高实时的搜索与数据分析引擎。它能很方便的使大量数据具有搜索、分析和探索的能力。充分利用 ElasticSearch 的水平伸缩性，能使数据在生产环境变得更有价值。它提供可扩展的搜索，Elasticsearch 是分布式的，这意味着索引可以被分成分片，每个分片可以有 0 个或多个副本，每个节点托管一个或多个分片，并充当协调器将操作委托给正确的分片。相关数据通常存储在同一个索引中，该索引由一个或多个主分片和零个或多个副本分片组成。

官方声称 Elasticsearch 的分布式特性，可以使它可以扩展至数百台（甚至数千台）服务器，并处理 PB 量级的数据。作为 Apache 许可条款下的开源项目，Elasticsearch 提供基于 RESTful web 接口。

索引（Index）

ElasticSearch 把数据存放到一个或者多个索引（indices）中，索引相当于关系型数据库中的库（Database）。

类型（Type）

类型只是对文档的逻辑分类，以便存取不同数据结构的文档，类型相当于关系型数据库中的表（Table），但是新版本的 ElasticSearch 已经弱化了这个概念，推荐一个索引只使用默认的一个类型：\_doc。

文档（Document）

索引存放和读取的基本单元就是文档，文档相当于关系型数据库中的一条记录，一个文档就是一个 JSON 对象，并且支持丰富的基本数据类型、数组、对象、嵌套对象等等。

映射（mapping）

映射就像关系型数据库中的 schema ，描述了文档具有的数据结构、字段、每个字段的数据类型、分词器、是否能被搜索等属性。

节点（Node）

单独的一个 ElasticSearch 服务器实例称为一个节点，一个节点存储着一个或多个索引的分片数据。

集群（Cluster）

集群是多个 ElasticSearch 节点的集合。

分片（Shard）和副本（Replica）

分片即一个数据仓库，存储着某个索引的文档数据，一个索引可以指定创建一个或多个分片，每个分片可以指定创建零个或多个副本分片。

Elasticsearch 默认会把一个索引分成五个分片，每个分片默认一个副本，分片和副本被分配到集群内的各个节点里，

当集群中的节点增加或减少时，Elasticsearch 会自动的在各节点中迁移分片，使得数据仍然均匀分布在集群里，即一个索引的数据被分成了多份并保存在不同的节点上，

Elasticsearch 会自动尽可能把一个索引的不同分片存储在不同的主机上，分片的副本也会尽可能存在不同的主机上，这样可以提高容错率，从而提高可用性，

如果其中部分节点挂了，Elasticsearch 会在其他节点读取到这个节点的分片的副本，不会影响数据的完整性。

以下是四个节点、两个分片、每个分片一个副本：

以下是三个节点、两个分片、每个分片两个副本：

健康状态（healthy）

green 绿色，这代表所有的主分片和副本分片都已分配，集群是 100% 可用的。

yellow 黄色，所有的主分片已经分片了，但至少还有一个副本是缺失的，不会有数据丢失，所以搜索结果是完整的，但高可用性在某种程度上被削弱。

red 红色，至少一个主分片以及它的全部副本都在缺失中，这意味着返回的数据是缺失的，而分配到这个分片上的写入请求会返回一个异常。

四、倒排索引原理

分析器（Analyzer）

分析器由这三部分组成：字符过滤器（Character filters）、分词器（Tokenizer）、词元过滤器（Token filters）。

字符过滤器（Character filters）

接收并处理原始文本，字符集转换、特殊符号过滤等等。

分词器（Tokenizer）

将字符串拆分成词元（Token），常用的内置分词器：标准分词器（Standard）、空格分词器（Whitespace），可安装第三方分词器：ik 分词器（ik\_smart、ik\_max\_word）。

词元过滤器（Token filters）

接收并处理词元，大小写转换、数据类型转换、纠错、英文单词单复数转换、近义词同义词转换。

分词

当一个文档被存储时，Elasticsearch 会使用分析器，依次执行字符过滤器、分词器、词元过滤器，从文档中提取出若干词元（Token）来支持索引的存储和搜索的过程就叫分词。

Standard 标准分词器是按英文单词进行分词，对中文是按单个汉字进行分词。

IK 中文分词器支持中文分词，有两种分词模式：ik\_max\_word 和 ik\_smart 模式

1. ik\_max\_word：会将文本做最细粒度的拆分，比如会将“中华人民共和国人民大会堂”拆分为“中华人民共和国、中华人民、中华、华人、人民共和国、人民、共和国、大会堂、大会、会堂等词语；

2. ik\_smart：会做最粗粒度的拆分，比如会将“中华人民共和国人民大会堂”拆分为中华人民共和国、人民大会堂。

倒排索引

现代搜索引擎的索引都是基于倒排索引，倒排索引也常被称为反向索引，被用来存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射。它是文档检索系统中最常用的数据结构。通过倒排索引，可以根据单词快速获取包含这个单词的文档列表。

假设一个文档集合包含以下五个文档，我们的任务就是对这个文档集合建立倒排索引：

文档 ID

文档内容

1

谷歌地图之父跳槽Facebook

2

谷歌地图之父加盟Facebook

3

谷歌地图创始人拉斯离开谷歌加盟Facebook

4

谷歌地图之父跳槽Facebook与Wave项目取消有关

5

谷歌地图之父拉斯加盟社交网站facebook

假设我们已经安装并使用了中文分词，对以上文档进行分词并建立最简单的倒排索引如下：

单词 ID

单词

倒排索引（DocID）

1

谷歌

1，2，3，4，5

2

地图

1，2，3，4，5

3

之父

1，2，4，5

4

跳槽

1，4

5

Facebook

1，2，3，4，5

6

加盟

2，3，5

7

创始人

3

8

拉斯

3，5

9

离开

3

10

与

4

11

Wave

4

12

项目

4

13

取消

4

14

有关

4

15

社交

5

16

网站

5

实际上倒排索引不仅会记录每个单词和对应的文档 ID，还会记录这个单词在对应文档中出现的频率，简称：词频（TF），这个指标在搜索结果排序中起了非常重要的作用：

单词 ID

单词

倒排索引（DocID:TF）

1

谷歌

(1:1)，(2:1)，(3:2)，(4:1)，(5:1)

2

地图

(1:1)，(2:1)，(3:1)，(4:1)，(5:1)

3

之父

(1:1)，(2:1)，(4:1)，(5:1)

4

跳槽

(1:1)，(4:1)

5

Facebook

(1:1)，(2:1)，(3:1)，(4:1)，(5:1)

6

加盟

(2:1)，(3:1)，(5:1)

7

创始人

(3:1)

8

拉斯

(3:1)，(5:1)

9

离开

(3:1)

10

与

(4:1)

11

Wave

(4:1)

12

项目

(4:1)

13

取消

(4:1)

14

有关

(4:1)

15

社交

(5:1)

16

网站

(5:1)

实际上倒排索引不仅会记录每个单词、对应的文档 ID、对应的词频，还会记录这个单词在文档中出现的位置、逆向文档频率，这些都将对搜索结果排序起非常重要的作用。

五、搜索排序算法

TF-IDF

在介绍 ES 计算文档得分之前，先来看一下 TF-IDF 算法。

TF-IDF（Term Frequency–Inverse Document Frequency）是一种用于信息检索与文本挖掘的常用加权算法。它是一种统计方法，用以评估一单词对于一个文档集或一个语料库中的其中一份文档的重要程度。单词的重要性随着它在文档中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。

TF-IDF 实际上是两个算法 TF 和 IDF 的乘积。

词频（Term Frequency，TF）

词频的所在对象是一个具体的文档，是指一个文档中出现某个单词（Term）的频率（Frequency）。这里用的是频率而不是次数，是为了防止文档内容过长从而导致某些单词出现过多。为了正确评价一个单词在一个文档中的重要程度，需要将出现次数归一化，其算法如下：

上面式子中 ni 是该词在文档中的出现次数，而分母 ∑nk=1nk 则是在档件中所有单词的出现次数之和。

逆向文档频率（Inverse Document Frequency，IDF）

逆向文档频率描述的对象是一个文档集合中，包含某个单词的文档数量。它表示的是一个单词在一个文档集合中的普遍重要程度。将其归一化的算法入下：

|D|：表示文档集合中的文档总数

|{j:ti∈dj}| ：包含单词ti的文档数目（即 ni≠0 的文档数目）如果单词不在数据中，就导致分母为零，因此一般情况下使用分母加了一个 1

最终 TF-IDF 即为：

某一特定文档内的高单词频率，以及该单词在整个文档集合中的低文档频率，可以产生出高权重的 TF-IDF。因此，TF-IDF 倾向于过滤掉常见的单词，保留重要的单词。

TF-IDF 总结

TF 表示的是一个单词在一段文本中的重要程度，随着单词的增加而增加

IDF 表示的是一个单词在一个文档集合中的重要程度，出现越普遍权重越低，越稀有权重越高，所以它随着包含该单词的文档数增加而降低

TF-IDF 算法举例

用上面的公式，计算一个例子。

假如一篇文档的总单词数是 100 个，而单词“学校”出现了 5 次，那么“学校”一单词在该文档中的词频（TF）就是

学校，一词在 1,000 份文档出现过，而文档总数是 1,000,000 份的话，其逆向文档频率就是

最后的 tf-idf 的分数为

OKapi BM25 算法原理

BM25（Best Match25）是在信息检索系统中根据提出的 query 对 document 进行评分的算法。

TF-IDF 算法是一个可用的算法，但并不太完美。它给出了一个基于统计学的相关分数算法，而 BM25 算法则是在此之上做出改进之后的算法。（为什么要改进呢？TF-IDF 不完美的地方在哪里？）

1. 当两篇描述“人工智能”的文档 A 和 B，其中 A 出现“人工智能” 100 次，B 出现“人工智能” 200 次。两篇文章的单词数量都是 10000，那么按照 TF-IDF 算法，A 的 TF 得分是：0.01，B 的 TF 得分是 0.02。得分上 B 比 A 多了一倍，但是两篇文章都是再说人工智能，TF 分数不应该相差这么多。可见单纯统计的 TF 算法在文本内容多的时候是不可靠的

2. 多篇文档内容的长度长短不同，对 TF 算法的结果也影响很大，所以需要将文本的长度也考虑到算法当中去

基于上面两点，BM25 算法做出了改进，最终该算法公式如下：

Q：文档集合

D：具体的文档

IDF(qi)：就是 TF-IDF 中的 IDF，表示单词 qi 在文档集合 Q 的 IDF 值

f(qi,D)：就是 TF-IDF 中的 TF，表示单词 qi 在文档 D 中的 TF 值

k1：词频饱和度（term frequency saturation）它用于调节饱和度变化的速率。它的值一般介于 1.2 到 2.0 之间。数值越低则饱和的过程越快速。（意味着两个上面 A、B 两个文档有相近的分数，因为他们都包含大量的“人工智能”这个单词）。在 ES 应用中为 1.2

b：文档长度归约，将文档的长度归约化到全部文档的平均长度，它的值在 0 和 1 之间，1 意味着全部归约化，0 则不进行归约化。在ES的应用中为 0.75

|D|：文档长度

avgdl：文本平均长度

总结，BM25 算法新增了以下常量来控制词频和文档长度对结果的影响：

词频饱和度（k1）：削弱因为词频（TF）的提升对权重得分的提升，k1 值越大，削弱效果越明显

文档长度归约（b）：是否将文档长度作为计算权重得分的因素，0 表示否，b 值越大，文档长度对权重得分的影响越大

文档的长度（|D|）：在文档长度归约（b）大于 0 的时候，文档长度越长，权重得分越低

Lucene 相关性算法

在了解了 TF-IDF 算法之后，再来了解 Lucene 中的相关性算法就很好理解了。

Lucene 中，相关性算法如下：

q：文档集合

d：具体的文档

t：单词

score(t, q, d)：表示包含查询单词 t 的文档 d 在文档集合 q 中的相关性得分

idf(t)：逆向文档频率，ES 中，逆向文档频率的算法是：

docCount：表示文档总数

docFreq：表示包含单词 t 的文档数量

boost(t)：查询时，指定的单词的权重，不指定时为 1

tfNorm(t, d)：词频，它用 BM25 替代了简单的 TF 算法，ES 中，其算法如下：

tfNorm(t, d)：单词t在文档 d 中的频率权重

f(t, d)：单词t在文档 d 中的出现次数

k1：词频饱和度，用于控制词频对结果的影响，数值越低则单词数量影响越小。它的值一般介于 1.2 到 2.0 之间。。在ES应用中为 1.2

b：文档长度归约，用于控制文档长度对结果的影响，数值越大文档长度影响越小。它的值在 0 和 1 之间，在ES的应用中为 0.75

|D|：文档 d 中查询该字段的文档长度

avgdl：文档集合中，所有查询该字段的平均长度

总结

ES 在搜索过程中，拿到文档 ID 之后，就会根据搜索词，计算每篇文档的相关性得分，用其进行排序，

对得分计算产生影响的变量有这些：

某个单词在一个文档中出现的频率：词频（TF）

包含某个单词的文档数量：逆向文档频率（IDF）

文档的长度（|D|）

所有文档的平均长度（avgdl）

单词权重（boost）

常量：词频饱和度（k1）

常量：文档长度归约（b）

实际搜索时计算会更复杂，涉及多字段的不同权重、多字段得分求和、模糊匹配、短语匹配、精确匹配、正则匹配、条件运算等等。

六、文档的 CRUD

创建文档

日期时间字段的值必须是 ISO8601 标准时间格式，才能被识别为 date 数据类型，或者通过手动定义日期时间格式。

curl -X PUT 'http://127.0.0.1:9200/course/\_doc/18399' \

--header 'Content-Type: application/json' \

--data-raw '{

"id": 18399,

"course\_title": "一年级语文系统测试班型丨系统课丨2021春",

"tags": ["标签1", "标签2", "标签3"],

"price":1000,

"created\_at":"2021-08-01T09:00:00+08:00",

"city": {

"config\_id": 344,

"key": "city",

"label": "其他城市",

"value": "122790",

"offline\_id": 18,

"online\_id": 344

},

"right": [

{

"config\_id": 78,

"key": "course\_right",

"label": "纸质讲义",

"value": "心田花开教研精心研发的精美纸质讲义，会免费为您邮寄到家。",

"offline\_id": 0,

"online\_id": 2

},

{

"config\_id": 239,

"key": "course\_right",

"label": "课前沟通",

"value": "辅导老师课前会电话、QQ、微信等全方位进行沟通，全面了解孩子情况后，为孩子提供更具针对性的提升建议。报名直播课程后，辅导老师会为每位学生提供辅导答疑服务。",

"offline\_id": 0,

"online\_id": 1

}

]

}'

{

"\_index": "course",

"\_type": "\_doc",

"\_id": "18399",

"\_version": 1,

"result": "created",

"\_shards": {

"total": 2,

"successful": 1,

"failed": 0

},

"\_seq\_no": 0,

"\_primary\_term": 1

}

查询文档

curl -X GET 'http://127.0.0.1:9200/course/\_doc/18399'

{

"\_index": "course",

"\_type": "\_doc",

"\_id": "18399",

"\_version": 1,

"\_seq\_no": 0,

"\_primary\_term": 1,

"found": true,

"\_source": {

"id": 18399,

"course\_title": "一年级语文系统测试班型丨系统课丨2021春",

"price":1000,

"created\_at": "2021-08-01T09:00:00+08:00",

"tags": [

"标签1",

"标签2",

"标签3"

],

"city": {

"config\_id": 344,

"key": "city",

"label": "其他城市",

"value": "122790",

"offline\_id": 18,

"online\_id": 344

},

"right": [

{

"config\_id": 78,

"key": "course\_right",

"label": "纸质讲义",

"value": "心田花开教研精心研发的精美纸质讲义，会免费为您邮寄到家。",

"offline\_id": 0,

"online\_id": 2

},

{

"config\_id": 239,

"key": "course\_right",

"label": "课前沟通",

"value": "辅导老师课前会电话、QQ、微信等全方位进行沟通，全面了解孩子情况后，为孩子提供更具针对性的提升建议。报名直播课程后，辅导老师会为每位学生提供辅导答疑服务。",

"offline\_id": 0,

"online\_id": 1

}

]

}

}

搜索文档

curl -X POST 'http://127.0.0.1:9200/course/\_search' \

--header 'Content-Type: application/json' \

--data-raw '{

"\_source":[

"id", "course\_title", "price", "created\_at"

],

"query":{

"match":{

"course\_title":"语文"

}

},

"script\_fields":{

"discount\_price":{

"script":{

"lang":"painless",

"source":"doc['\''price'\''].value \* 0.8"

}

}

},

"docvalue\_fields":[

{

"field":"created\_at",

"format":"date"

}

],

"highlight":{

"fields":{

"course\_title":{

"pre\_tags":"<em>",

"post\_tags":"</em>"

}

}

},

"sort":[

{

"price":{

"order":"desc"

}

},

{

"id":{

"order":"desc"

}

}

],

"from":0,

"size":10

}'

{

"took": 3,

"timed\_out": false,

"\_shards": {

"total": 5,

"successful": 5,

"skipped": 0,

"failed": 0

},

"hits": {

"total": 1,

"max\_score": null,

"hits": [

{

"\_index": "course",

"\_type": "\_doc",

"\_id": "18399",

"\_score": null,

"\_source": {

"price": 1000,

"course\_title": "一年级语文系统测试班型丨系统课丨2021春",

"created\_at": "2021-08-01T09:00:00+08:00",

"id": 18399

},

"fields": {

"created\_at": [

"2021-08-01"

],

"discount\_price": [

800.0

]

},

"highlight": {

"course\_title": [

"一年级<em>语</em><em>文</em>系统测试班型丨系统课丨2021春"

]

},

"sort": [

1000,

18399

]

}

]

}

}

更新文档

每次更新一条文档的时候，全量更新整条文档的所有字段，重新建立文档索引

curl -X PUT 'http://127.0.0.1:9200/course/\_doc/18399' \

--header 'Content-Type: application/json' \

--data-raw '{

"id": 18399,

"course\_title": "一年级语文系统测试班型丨系统课丨2021春",

"tags": ["标签1", "标签2", "标签3"],

"price":1000,

"created\_at":"2021-08-01T09:00:00+08:00",

"city": {

"config\_id": 344,

"key": "city",

"label": "其他城市",

"value": "122790",

"offline\_id": 18,

"online\_id": 344

},

"right": [

{

"config\_id": 78,

"key": "course\_right",

"label": "纸质讲义",

"value": "心田花开教研精心研发的精美纸质讲义，会免费为您邮寄到家。",

"offline\_id": 0,

"online\_id": 2

},

{

"config\_id": 239,

"key": "course\_right",

"label": "课前沟通",

"value": "辅导老师课前会电话、QQ、微信等全方位进行沟通，全面了解孩子情况后，为孩子提供更具针对性的提升建议。报名直播课程后，辅导老师会为每位学生提供辅导答疑服务。",

"offline\_id": 0,

"online\_id": 1

}

]

}'

{

"\_index": "course",

"\_type": "\_doc",

"\_id": "18399",

"\_version": 5,

"result": "updated",

"\_shards": {

"total": 2,

"successful": 1,

"failed": 0

},

"\_seq\_no": 4,

"\_primary\_term": 1

}

部分更新

curl -X POST 'http://127.0.0.1:9200/course/\_doc/18399/\_update' \

--header 'Content-Type: application/json' \

--data-raw '{

"doc":{

"price":800

}

}'

curl -X POST 'http://127.0.0.1:9200/course/\_doc/18399/\_update' \

--header 'Content-Type: application/json' \

--data-raw '{

"script":{

"lang":"painless",

"source":"ctx.\_source.price += 100"

}

}'

{

"\_index": "course",

"\_type": "\_doc",

"\_id": "18399",

"\_version": 13,

"result": "updated",

"\_shards": {

"total": 2,

"successful": 1,

"failed": 0

},

"\_seq\_no": 12,

"\_primary\_term": 1

}

批量更新

批量更新是基于文档的快照，如果文档在生成快照和处理索引请求之间发生更改，更新时将遇到版本冲突，

所有更新和查询失败都会导致操作中止，已执行的更新仍然存在，进程不会回滚，只会中止。也可以主动设置 conflicts=proceed 遇到文档版本冲突时继续。

curl -X POST 'http://127.0.0.1:9200/course/\_update\_by\_query' \

--header 'Content-Type: application/json' \

--data-raw '{

"script":{

"lang":"painless",

"source":"ctx.\_source.price += 100"

},

"query":{

"terms":{

"id":[18400, 18401]

}

}

}'

{

"took": 17,

"timed\_out": false,

"total": 2,

"updated": 2,

"deleted": 0,

"batches": 1,

"version\_conflicts": 0,

"noops": 0,

"retries": {

"bulk": 0,

"search": 0

},

"throttled\_millis": 0,

"requests\_per\_second": -1.0,

"throttled\_until\_millis": 0,

"failures": []

}

删除文档

curl -X DELETE 'http://127.0.0.1:9200/course/\_doc/18400'

{

"\_index": "course",

"\_type": "\_doc",

"\_id": "18400",

"\_version": 6,

"result": "deleted",

"\_shards": {

"total": 2,

"successful": 1,

"failed": 0

},

"\_seq\_no": 5,

"\_primary\_term": 1

}

批量删除

跟批量更新一样，可能会遇到版本冲突和操作终止的问题。

curl -X POST 'http://127.0.0.1:9200/course/\_doc/\_delete\_by\_query' \

--header 'Content-Type: application/json' \

--data-raw '{

"query":{

"terms":{

"id":[18400, 18401]

}

}

}'

{

"took": 22,

"timed\_out": false,

"total": 2,

"deleted": 2,

"batches": 1,

"version\_conflicts": 0,

"noops": 0,

"retries": {

"bulk": 0,

"search": 0

},

"throttled\_millis": 0,

"requests\_per\_second": -1.0,

"throttled\_until\_millis": 0,

"failures": []

}

删除索引

curl -X DELETE 'http://127.0.0.1:9200/course'

{

"acknowledged": true

}