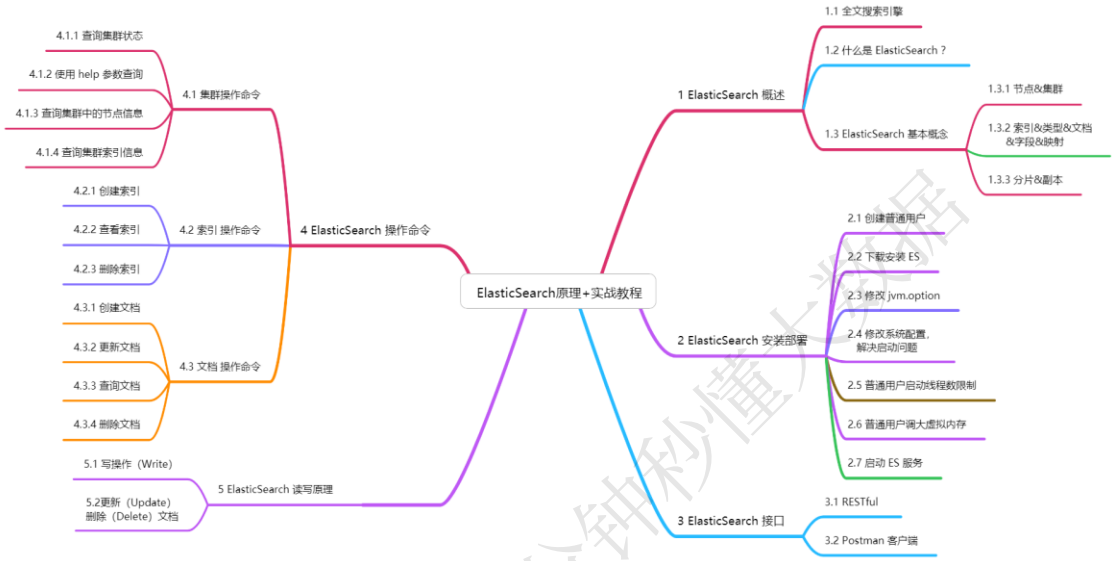
大家好，我是土哥

今天为大家带来流计算领域经常使用的组件 ElasticSearch，本文通过原理+实战操教程带领大家快速学会 ElasticSearch 搜索引擎，以下内容全部经过实战操作，可以根据文档进行学习~

大纲如下：



文档来自：3分钟秒懂大数据

1 ElasticSearch 概述

1.1 全文搜索引擎

小伙伴们经常使用 google 或者百度进行搜索内容，在输入框中输入**关键字**，这个时候，网站会将包含 **关键字** 的所有网页返回，大家有没有想过，**为什么输入关键字就可以查到结果呢？**

同时网站上返回的页面内容大多都是一些 **非结构化** 的文本数据，对于大量的文本数据，肯定是不会存储到**数据库**中的，原因如下：

（1）非结构化文本数据，关系型数据库搜索不能很好支持全文索引扫描整张表。

（2）查询效率低下，即使对 SQL 进行大量优化，其效果也收效甚微。

（3）对于 insert 和 update 操作都会重新构建索引，维护非常麻烦。

针对上述问题，在生产环境中，面对海量的数据，若想要**毫秒级**查询到结构化数据或非结构化数据，我们就需要专业，健壮，强大的全文搜索引擎。

全文搜索引擎的 **工作原理**：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。这个过程类似于通过字典中的检索字表查字的过程

1.2 什么是 ElasticSearch ？

为了实现毫秒级的搜索效果，ElasticSearch 出现了~

ElasticSearch（弹性搜索）：是一款开源的分布式、RESTful 风格的搜索和数据分析引擎，它底层基于 Apache Lucene 开源库进行封装，其不仅仅提供分布式多用户能力的全文搜索引擎，还可以被准确形容为：

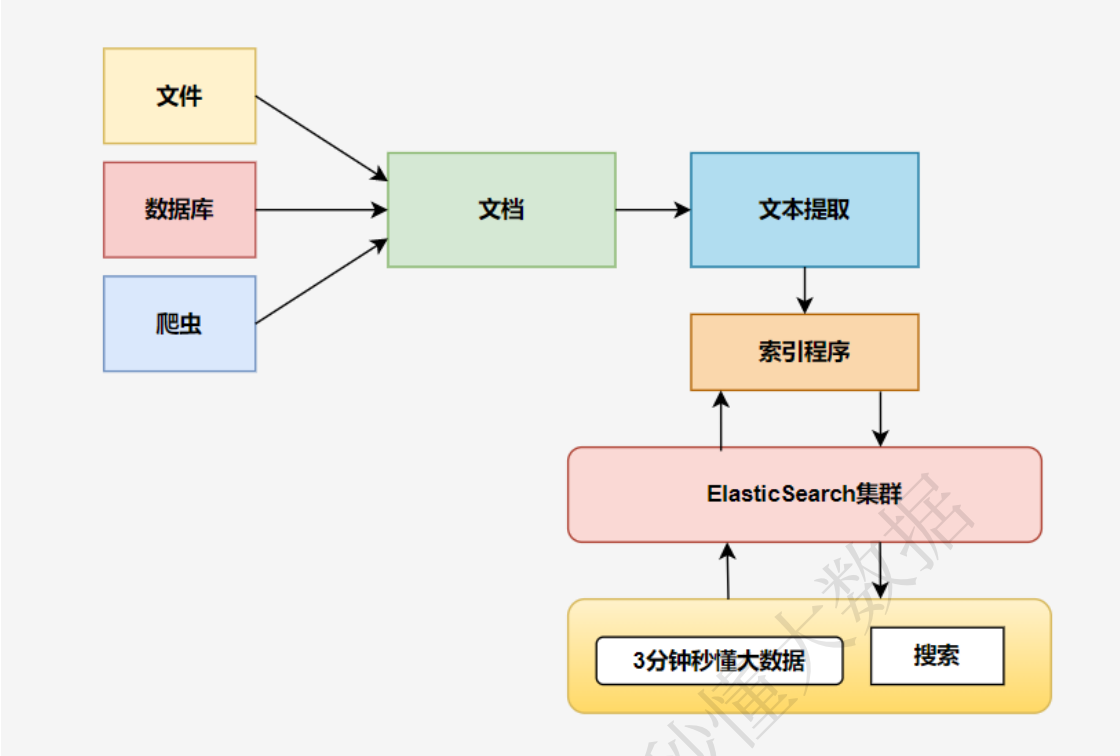
1、一个分布式的实时文档存储，每个字段可以被**索引**与**搜索**；

2、一个分布式**实时**分析搜索引擎；

3、能胜任**上百个**节点的扩展，并支持 PB 级别结构化和非结构化数据。

ElasticSearch 搜索整体架构如下图所示：

文章来自公众号：3分钟秒懂大数据



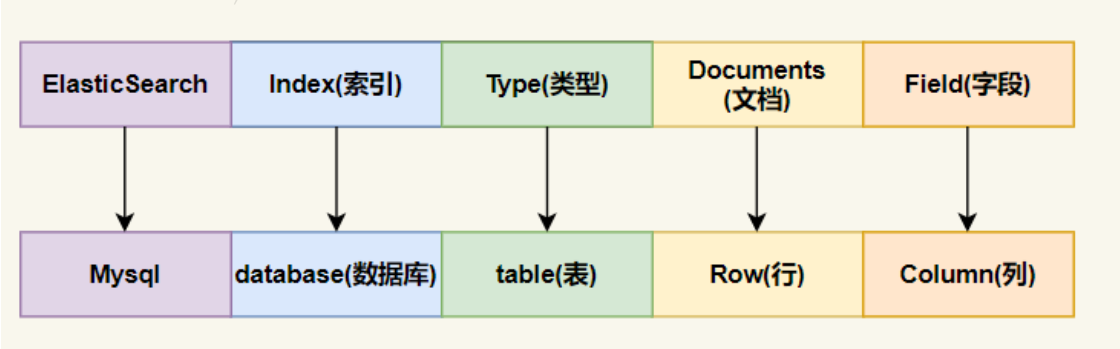
1.3 ElasticSearch 基本概念

1.3.1 节点&集群

Elasticsearch 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 Elasticsearch 实例。单个 Elasticsearch 实例称为一个节点（Node），一组节点构成一个集群（Cluster）。

1.3.2 索引&类型&文档&字段&映射

映射关系如下图：



什么是 index （索引） ？

一个 **索引** 就是一个拥有几分相似特征的文档的集合。ES 将数据存储于一个或多个索引中，**索引** 就相当于 SQL 中的一个**数据库**。

什么是 Type(类型)？

类型是索引内部的逻辑分区（category/partition）,然而其意义完全取决于用户需求。因此，一个索引内部可定义一个或多个类型（type）。一般来说，类型就是为那些拥有相同的域的文档做的预定义。类比传统的关系型数据库领域来说，**类型** 相当于 **表**，7.x 版本默认使用 _doc 作为 type 。

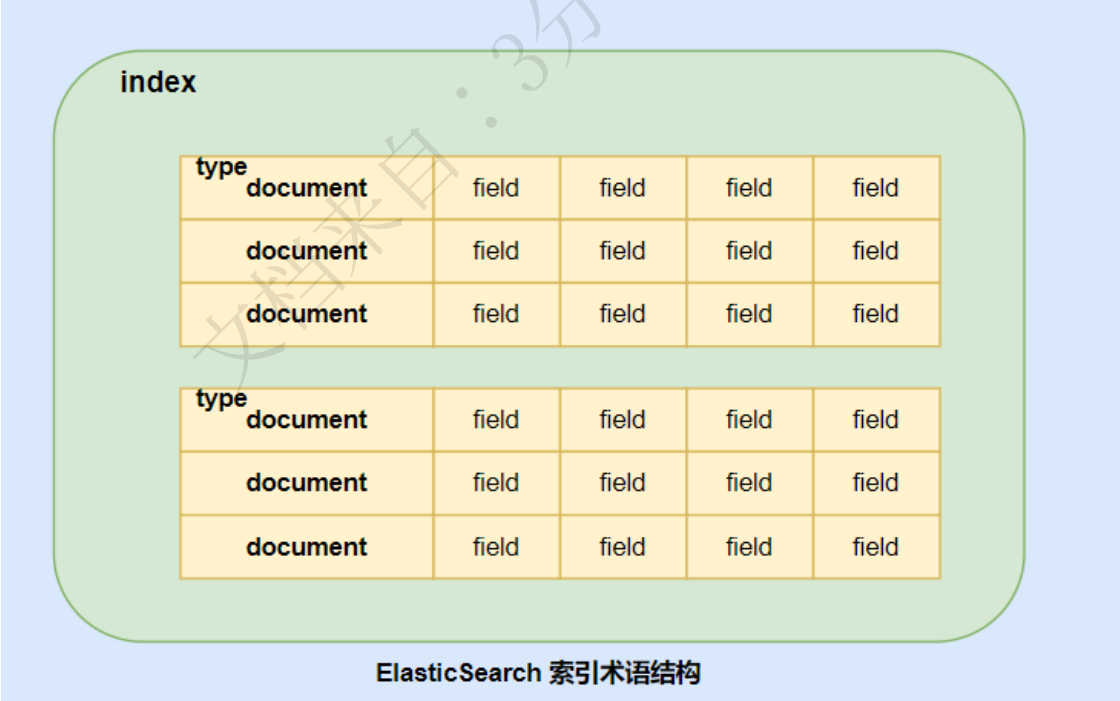
什么是 Document(文档)？

文档是 Lucene 索引和搜索的 **原子单位**，它是包含了一个或多个域的容器，基于 **Json** 格式进行表示。文档有一个或多个域组成，每个域拥有一个名字及一个或多个值，有多个值的域通常被称为 **多值域**，每个文档可以存储不同的域集，但同一类型下的文档至应该有某种程度上的相似之处。相当于 **mysql** 表中的 **row** 。

什么是 Field (字段)？

Field 是相当于数据库中的 Column

上述索引&类型&文档&字段结构图如下：



ElasticSearch 索引术语结构

文档来自：3分钟秒懂大数据

什么是 Mapping(映射)?

Mapping 是定义文档及其包含的字段如何存储和索引的过程。**Mapping** 是 **ES** 中的一个很重要的内容，它类似于传统关系型数据中 table 的 schema，用于定义一个索引（index）的某个类型（type）的数据结构。

1.3.3 分片&副本

什么是 Shard (分片)?

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。

分片之所以重要，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
3. 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

什么是 Replica (副本)?

副本是一个分片的精确复制，每个分片可以有零个或多个副本。副本的作用：

1. 提高系统的容错性，当某个节点某个分片损坏或丢失时，可以从副本中恢复。
2. 提高 ES 查询效率，ES 会自动对搜索请求进行负载均衡。

2 ElasticSearch 安装部署

2.1 创建普通用户

```
#1 创建普通用户名，密码
[root@hlink1 lyz]# useradd lyz
[root@hlink1 lyz]# passwd lyz

#2 然后 关闭 xshell  重新登录 ip 地址  用 lyz 用户登录

#3 为 lyz 用户分配 sudoer 权限
[lyz@hlink1 ~]$ su
```

```
[lyz@hlink1 ~]$ vi /etc/sudoers
# 在 root  ALL=(ALL)    ALL 下面添加普通用户权限
     lyz  ALL=(ALL)    ALL
```

2.2 下载安装 ES

```
# 4 下载安装包
[lyz@hlink1 ~]$ wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.15.2-linux-x86_64.tar.gz
# 5 解压安装包
[lyz@hlink1 ~]$ tar -xzf elasticsearch-7.15.2-linux-x86_64.tar.gz

# 6 修改配置
[lyz@hlink1 ~]# cd elasticsearch-7.15.2/config
[lyz@hlink1 elasticsearch-7.15.2]# mkdir log
[lyz@hlink1 elasticsearch-7.15.2]# mkdir data
[lyz@hlink1 elasticsearch-7.15.2]# cd config
[lyz@hlink1 config]# rm -rf elasticsearch.yml
[lyz@hlink1 config]# vim elasticsearch.yml

# 粘贴如下内容

# 配置集群名称，保证每个节点的名称相同，如此就能都处于一个集群之内了
cluster.name: lyz-es
# # 每一个节点的名称，必须不一样
node.name: hlink1
path.data: /home/lyz/elasticsearch-7.15.2/log
path.logs: /home/lyz/elasticsearch-7.15.2/data
network.host: 0.0.0.0
# # http 端口（使用默认即可）
http.port: 9200
# # 集群列表,你 es 集群的 ip 地址列表
discovery.seed_hosts: ["hlink1"]
# # 启动的时候使用一个 master 节点
cluster.initial_master_nodes: ["hlink1"]
bootstrap.system_call_filter: false
bootstrap.memory_lock: false
http.cors.enabled: true
http.cors.allow-origin: "*"
```

2.3 修改 jvm.option

修改 jvm.option 配置文件，调整 jvm 堆内存大小,每个人根据自己服务器的内存大小来进行调整

```
# 7 修改 jvm.option 配置文件
[lyz@hlink1 config]# vim jvm.options
```

```
-Xms2g
-Xmx2g
```

2.4 修改系统配置，解决启动问题

由于使用普通用户来安装 es 服务，且 es 服务对服务器的资源要求比较多，包括内存大小，线程数等。所以我们需要给普通用户解开资源的束缚

ES 因为需要大量的创建索引文件，需要大量的打开系统的文件，所以我们需要解除 linux 系统当中打开文件最大数目的限制，不然 ES 启动就会抛错

进入 Root 用户

```
# 8 进入 root 用户
[lyz@hlink1 config]# su
Password:

# 9 在最下面添加如下内容：注意*不要去掉了
[root@hlink1 config]# sudo vim /etc/security/limits.conf

* soft nofile 65536
* hard nofile 131072
* soft nproc 2048
* hard nproc 4096
```

2.5 普通用户启动线程数限制

修改普通用户可以创建的最大线程数

```
10 若为 Centos7,执行下面的命令
[root@hlink1 config]# sudo vim /etc/security/limits.d/20-nproc.conf

# 找到如下内容：
* soft nproc 1024#修改为
* soft nproc 4096
```

2.6 普通用户调大虚拟内存

```
# 11 调大系统的虚拟内存
[root@hlink1 config]# vim /etc/sysctl.conf

vm.max_map_count=262144

# 12 执行 sysctl -p
# 行完了 sysctl -p 若输出的结果和你配置的一样,说明配置成功了.
[root@hlink1 config]# sysctl -p
vm.max_map_count = 262144
```

文章来自公众号：3分钟秒懂大数据

2.7 启动 ES 服务

```
# 13 切换用户
[root@hlink1 config]# exit
exit
[lyz@hlink1 config]$

# 直接启动 es 或者 后台启动 es
[lyz@hlink1 config]$ cd ..
[lyz@hlink1 elasticsearch-7.15.2]$ cd bin
# 直接启动
[lyz@hlink1 bin]$ ./elasticsearch
# 后台启动 nohup ./elasticsearch 2>&1 &

# 浏览器访问 http://hlink1:9200/?pretty
```


```
{
    name: "hlink1",
    cluster_name: "lyz-es",
    cluster_uuid: "czr5hb9_QJKsfsHYEYdPpw",
  - version: {
        number: "7.15.2",
        build_flavor: "default",
        build_type: "tar",
        build_hash: "93d5a7f6192e8a1a12e154a2b81bf6fa7309da0c",
        build_date: "2021-11-04T14:04:42.515624022Z",
        build_snapshot: false,
        lucene_version: "8.9.0",
        minimum_wire_compatibility_version: "6.8.0",
        minimum_index_compatibility_version: "6.0.0-beta1"
    },
    tagline: "You Know, for Search"
}
```


3 ElasticSearch 接口

3.1 RESTful

ElasticSearch 对外提供的 API 是以 HTTP 协议的方式，通过 JSON 格式以 REST 约定对外提供。

HTTP 配置文件放在 elasticsearch.yml 中。REST 通常是开发的一种约定，当所有开发者都遵循这种约定时，就会简化开发的沟通成本。

REST 约定用 HTTP 的请求头 POST、GET、PUT、DELETE 正好对应 CRUD（Create、Read、Update、Delete）四种数据操作。如果应用程序符合 REST 原则，可称为 RESTful Web Service。

REST 请求头

HTTP 方法	数据处理	说明
POST	Create	新增资源
GET	Read	读取资源
PUT	Update	更新资源
DELETE	Delete	删除资源

文档来自：3分钟秒懂大数据

3.2 Postman 软件安装

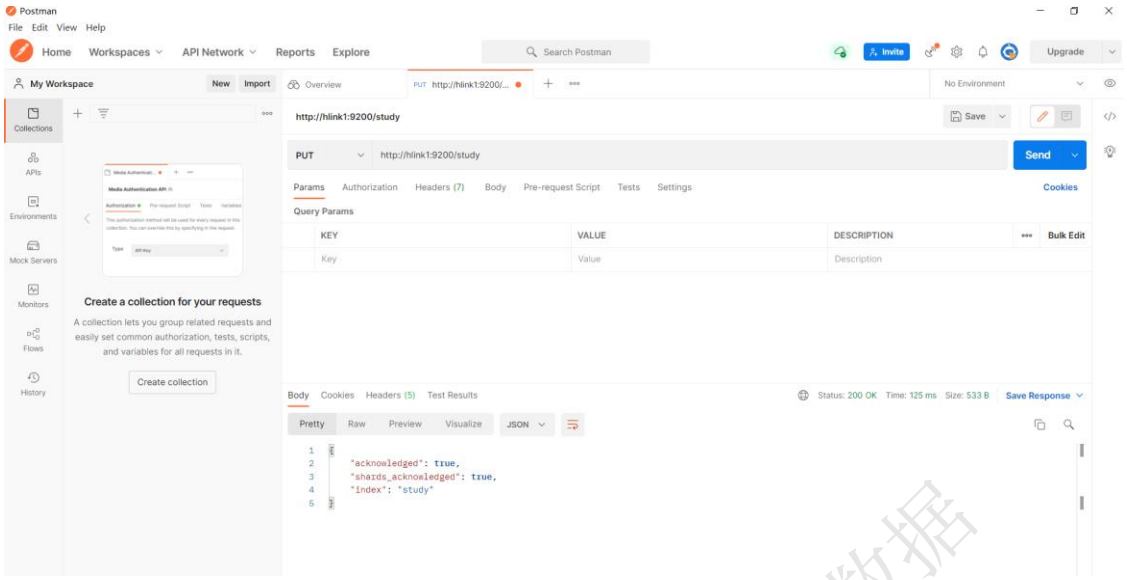
本文使用 Postman 作为 Elasticsearch 的客户端进行连接使用

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。

直接在官网进行下载安装即可：

Postman 下载：https://www.getpostman.com/apps

安装完之后进行简单的注册



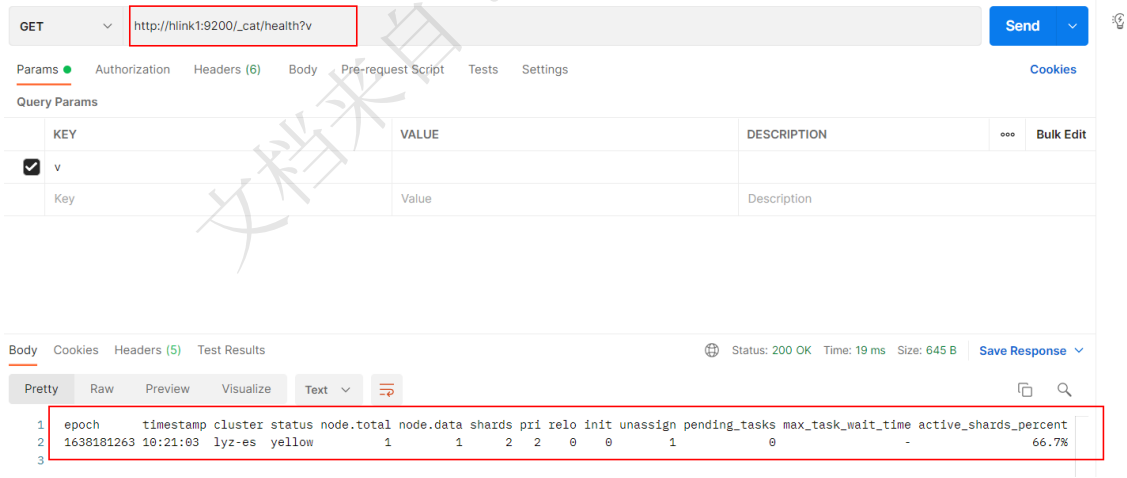
4 ElasticSearch 操作命令

4.1 集群信息操作命令

4.1.1 查询集群状态

（1）使用 Postman 客户端直接向 ES 服务器发 GET 请求

```
http://hlink1:9200/_cat/health?v
```



（2）使用服务端进行查询

```
curl -XGET "hlink1:9200/_cat/health?v"
```

文档来自公众号：3分钟秒懂大数据

```
epoch      timestamp cluster status node.total node.data shards pri rel
o init unassign pending_tasks max_task_wait_time active_shards_percent
1638181746 10:29:06  lyz-es  yellow          1         1      2   2
0    0        1             0                  -                66.7%
```

返回结果的主要字段意义：

- cluster：集群名，是在 ES 的配置文件中配置的 cluster.name 的值。
- status：集群状态。集群共有 green、yellow 或 red 中的三种状态。green 代表一切正常（集群功能齐全），yellow 意味着所有的数据都是可用的，但是某些复制没有被分配（集群功能齐全），red 则代表因为某些原因，某些数据不可用。如果是 red 状态，则要引起高度注意，数据很有可能已经丢失。
- node.total：集群中的节点数。
- node.data：集群中的数据节点数。
- shards：集群中总的分片数量。
- pri：主分片数量，英文全称为 private。
- relo：复制分片总数。
- unassign：未指定的分片数量，是应有分片数和现有的分片数的差值（包括主分片和复制分片）。

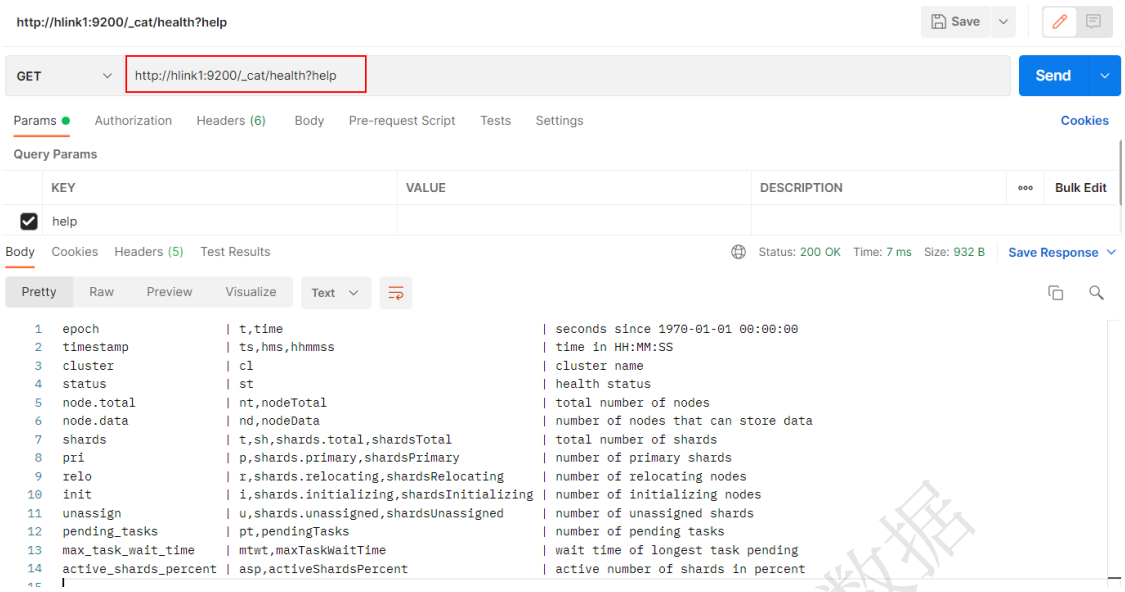
4.1.2 使用 help 参数查询

（1）使用 Postman 客户端查询

```
http://hlink1:9200/_cat/health?help
```

在请求中添加 help 参数来查看每个操作返回结果字段的意义。

文章来自公众号：3分钟秒懂大数据



（2）使用 服务端 查询

```
curl -XGET "hlink1:9200/_cat/health?help"
```

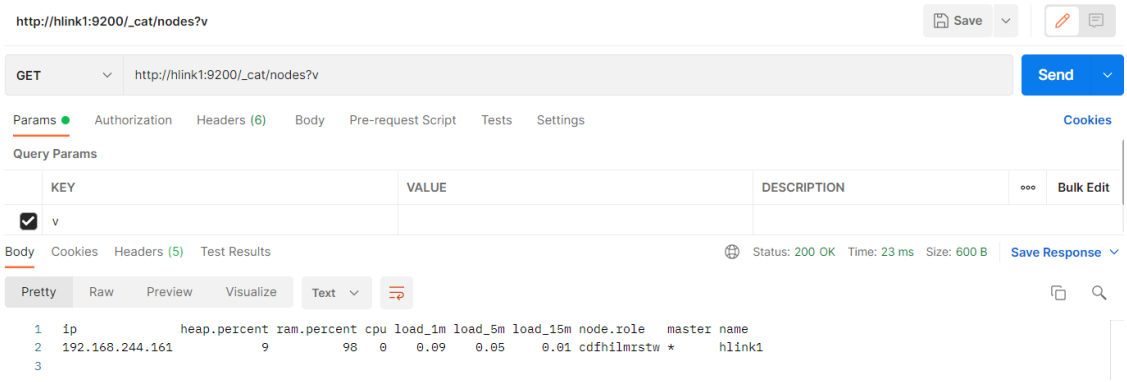
```
[lyz@hlink1 config]$ curl -XGET "hlink1:9200/_cat/health?help"
epoch                | t,time                                | seconds since 1970-01-01 00:00:00
timestamp            | ts,hms,hhmmss                         | time in HH:MM:SS
cluster              | cl                                    | cluster name
status               | st                                    | health status
node.total           | nt,nodeTotal                          | total number of nodes
node.data            | nd,nodeData                           | number of nodes that can store data
shards               | t,sh,shards.total,shardsTotal         | total number of shards
pri                  | p,shards.primary,shardsPrimary        | number of primary shards
relo                 | r,shards.relocating,shardsRelocating  | number of relocating nodes
init                 | i,shards.initializing,shardsInitializing | number of initializing nodes
unassign             | u,shards.unassigned,shardsUnassigned  | number of unassigned shards
pending_tasks        | pt,pendingTasks                       | number of pending tasks
max_task_wait_time   | mtwt,maxTaskWaitTime                  | wait time of longest task pending
active_shards_percent | asp,activeShardsPercent              | active number of shards in percent
```

```
# 指定返回的参数
curl -XGET "hlink1:9200/_cat/health?h=cluster,pri,relo&v"
```

4.1.3 查询集群中的节点信息

（1）使用 Postman 客户端查询

```
http://hlink1:9200/_cat/nodes?v
```



（2）使用服务端查询

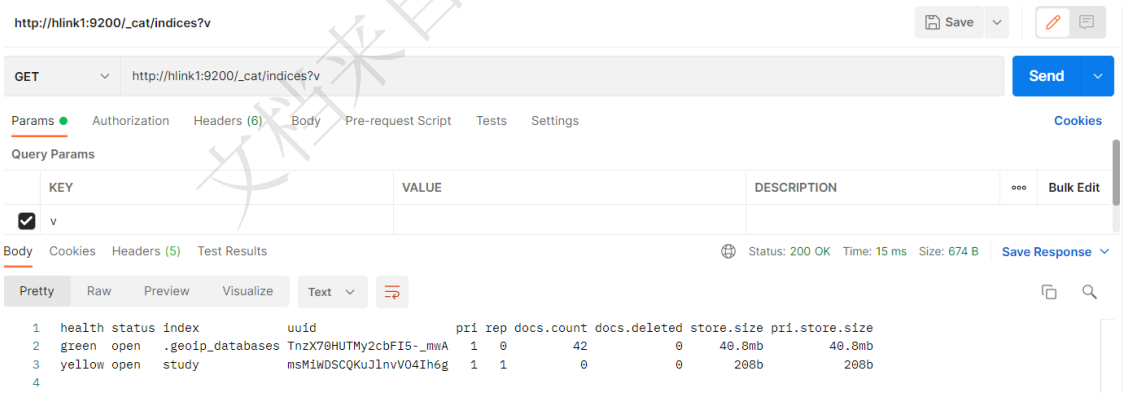
```
curl -XGET "hlink1:9200/_cat/nodes?v"
```

```
[lyz@hlink1 config]$ curl -XGET "hlink1:9200/_cat/nodes?v"
ip              heap.percent ram.percent cpu load_1m load_5m load_15m node.role   master name
192.168.244.161            7          98   0    0.11    0.06     0.01 cdfhilmrstw *      hlink1
[lyz@hlink1 config]$
```

4.1.4 查询集群索引信息

（1）使用 Postman 客户端查询

```
http://hlink1:9200/_cat/indices?v
```



（2）使用服务端查询

```
curl -XGET "hlink1:9200/_cat/indices?v"
```

```
[lyz@hlink1 config]$ curl -XGET "hlink1:9200/_cat/indices?v"
health status index           uuid                   pri rep docs.count docs.deleted store.size pri.store.size
green  open   .geoip_databases TnzX70HUTMy2cbFI5-_mwA   1   0         42            0     40.8mb         40.8mb
yellow open   study           msMiWDSCQKuJlnvVO4Ih6g   1   1          0            0       208b           208b
[lyz@hlink1 config]$
```

4.2 index 操作命令

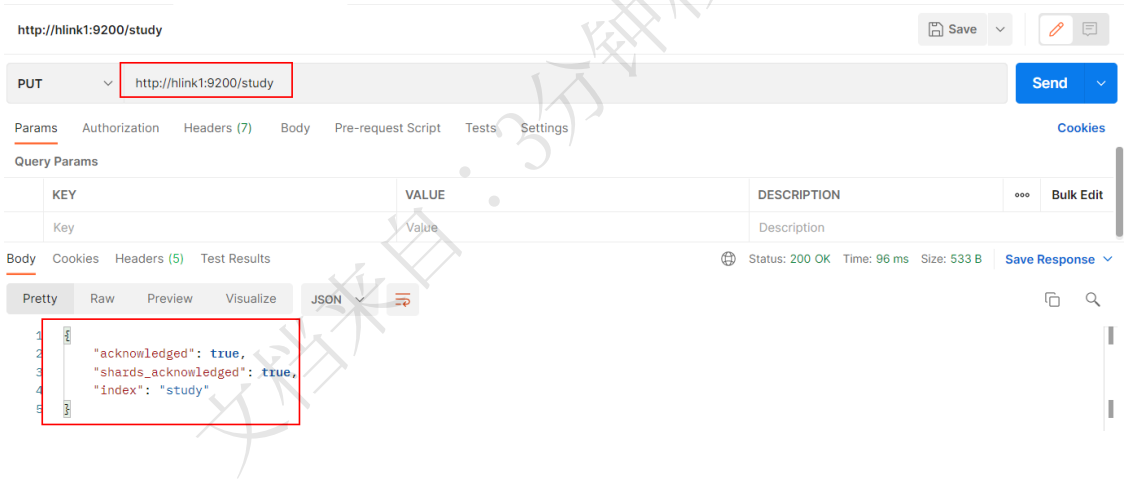
后面操作命令全部使用 Postman 客户端进行操作，想查看详细文档，在公众号：【3 分钟秒懂大数据】，回复：【elasticsearch】，获取更详细文档。

4.2.1 创建索引

在 Postman 中，向 ES 服务器发 PUT 请求

```
http://hlink1:9200/study
```

```
{
 "acknowledged"【响应结果】: true, # true 操作成功
 "shards_acknowledged"【分片结果】: true, # 分片操作成功
 "index"【索引名称】: "study"
}
```



4.2.2 查看索引

在 Postman 中，向 ES 服务器发 GET 请求

```
http://hlink1:9200/study
```

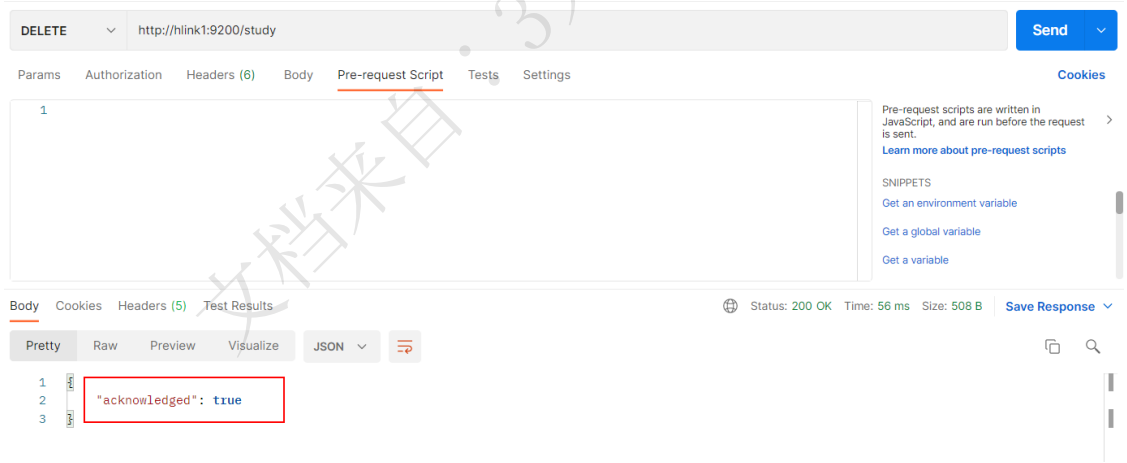
```
{
    "study": {
        "aliases": {},
        "mappings": {},
        "settings": {
```

```
            "index": {
                "routing": {
                    "allocation": {
                        "include": {
                            "_tier_preference": "data_content"
                        }
                    }
                },
                "number_of_shards": "1",
                "provided_name": "study",
                "creation_date": "1638183519598",
                "number_of_replicas": "1", #分片数量
                "uuid": "dg7HnAAiQEeDMJ7kMtA2Qw",
                "version": {
                    "created": "7150299"
                }
            }
        }
    }
}
```

4.2.3 删除索引

在 Postman 中，向 ES 服务器发 DELETE 请求

```
http://hlink1:9200/study
```

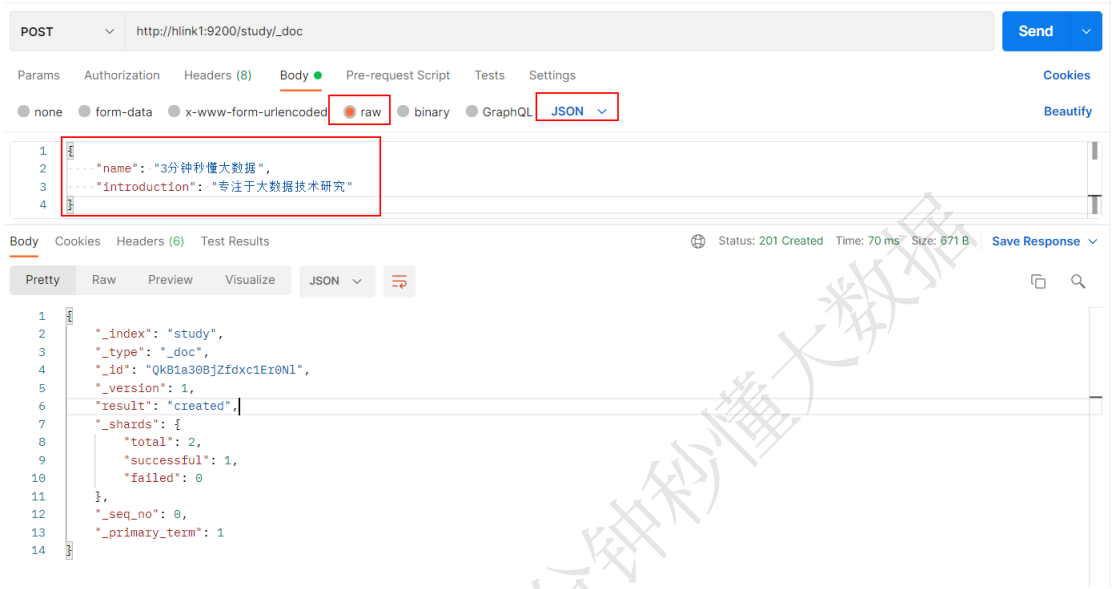


4.3 document 操作命令

4.3.1 创建文档

索引创建完成后，可以创建 document 文档，该文档对应数据库中 表的行数据 添加的数据格式为 JSON 内容

```
#请求体内容如下：
{
    "name": "3 分钟秒懂大数据",
    "introduction": "专注于大数据技术研究"
}
# 向 ES 服务器发 POST 请求
http://hlink1:9200/study/_doc
```

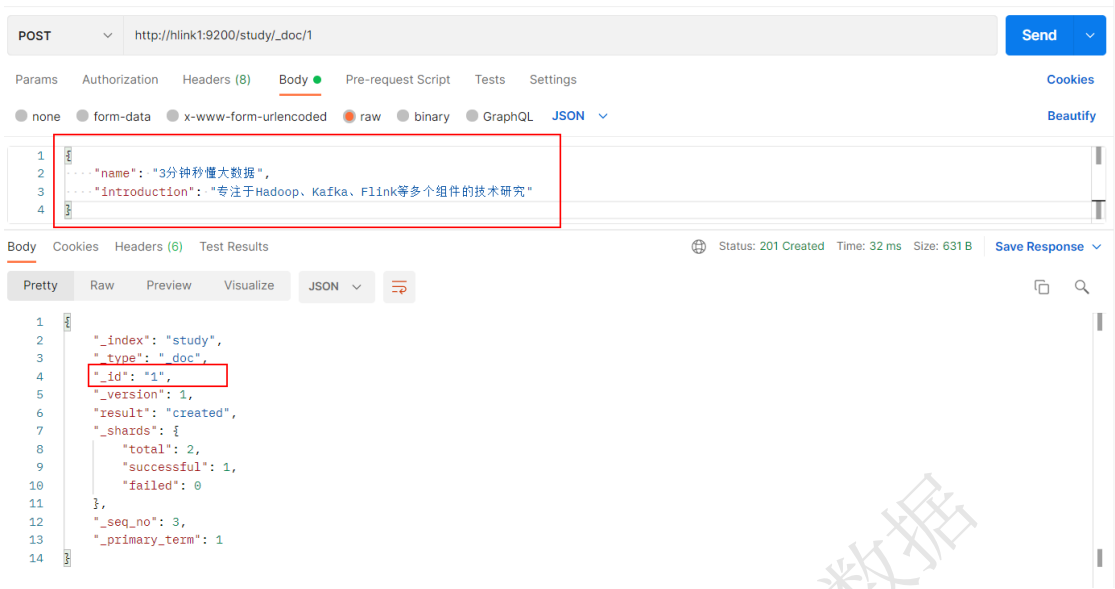


4.3.2 更新文档

上面的数据创建后，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。更新时可以指定唯一性标识

```
#请求体内容如下：
{
    "name": "3 分钟秒懂大数据",
    "introduction": "专注于 Hadoop、Kafka、Flink 等多个组件的技术研究"
}
# 向 ES 服务器发 POST 请求
http://hlink1:9200/study/_doc/1
```

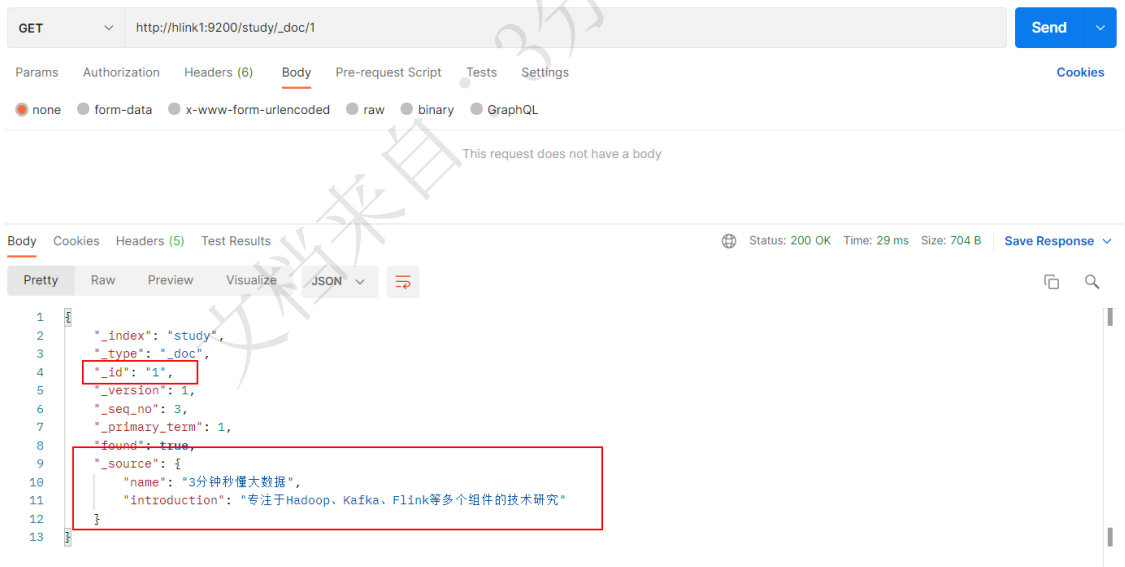
文档来自：3分钟秒懂大数据



4.3.3 查询文档

查看文档时，需要指明文档的唯一性标识

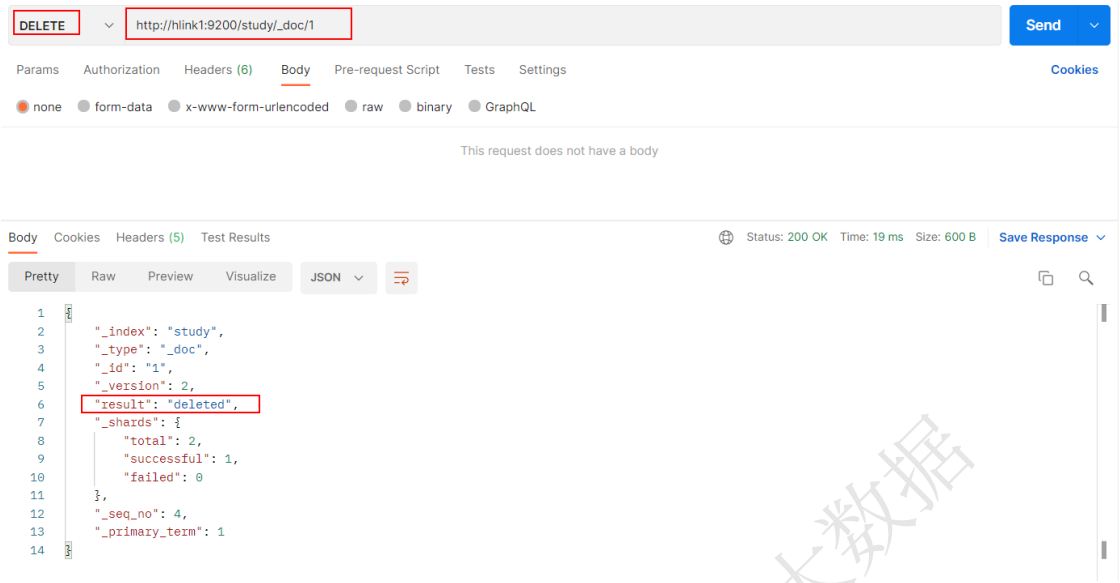
`http://hlink1:9200/study/_doc/1`



4.3.4 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）
Postman 向服务器发送 delete 请求

```
http://hlink1:9200/study/_doc/1
```



5 ElasticSearch 读写原理

5.1 写操作（Write）：针对文档的 CRUD 操作

索引新文档（Create）

当用户向一个节点提交了一个索引新文档的请求，节点会计算新文档应该加入到哪个分片（shard）中。每个节点都有每个分片存储在哪个节点的信息，因此协调节点会将请求发送给对应的节点。注意这个请求会发送给主分片，等主分片完成**索引**，会并行将请求发送到**其所有副本分片**，保证每个分片都持有最新数据。

每次写入新文档时，都会先写入 **内存** 中，并将这一操作写入一个 translog 文件（transaction log）中，此时如果执行搜索操作，这个新文档还不能被索引到。

文章来自公众号：3分钟秒懂大数据

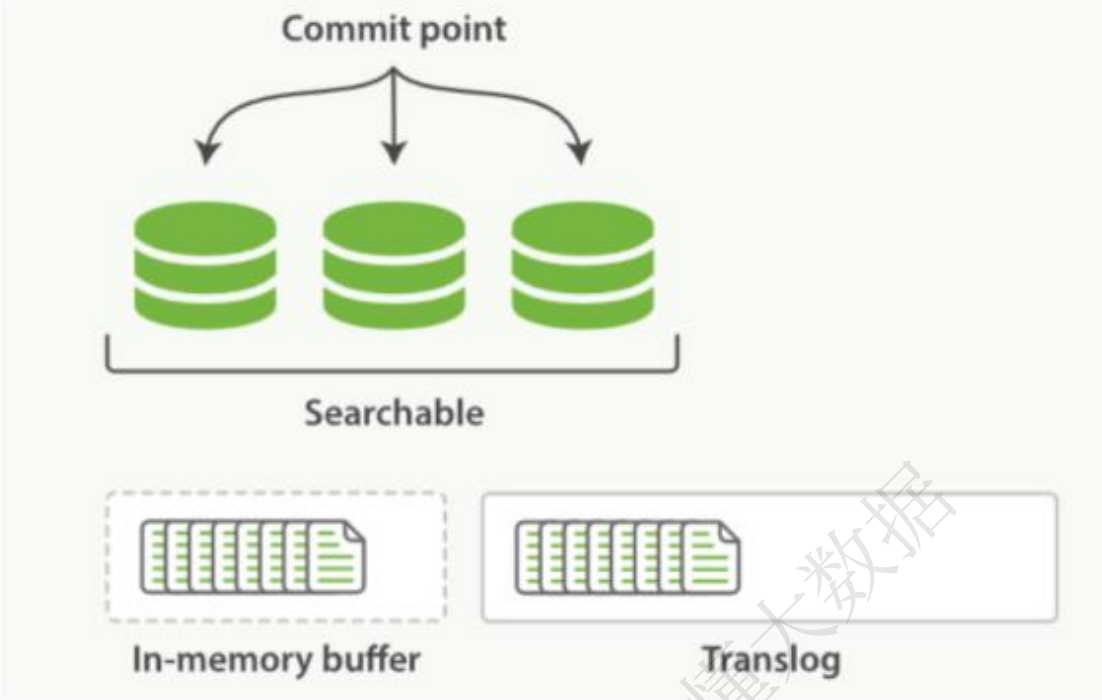


图1、新文档被写入内存，操作被写入translog

ES 会每隔 **1** 秒时间（这个时间可以修改）进行一次刷新操作（refresh），此时在这 1 秒时间内写入内存的新文档都会被写入一个文件系统缓存（`filesystem cache`）中，并构成一个分段（`segment`）。此时这个 segment 里的文档可以被搜索到，但是尚未写入硬盘，即如果此时发生断电，**则这些文档可能会丢失**。

文档来自公众号：3分钟秒懂大数据

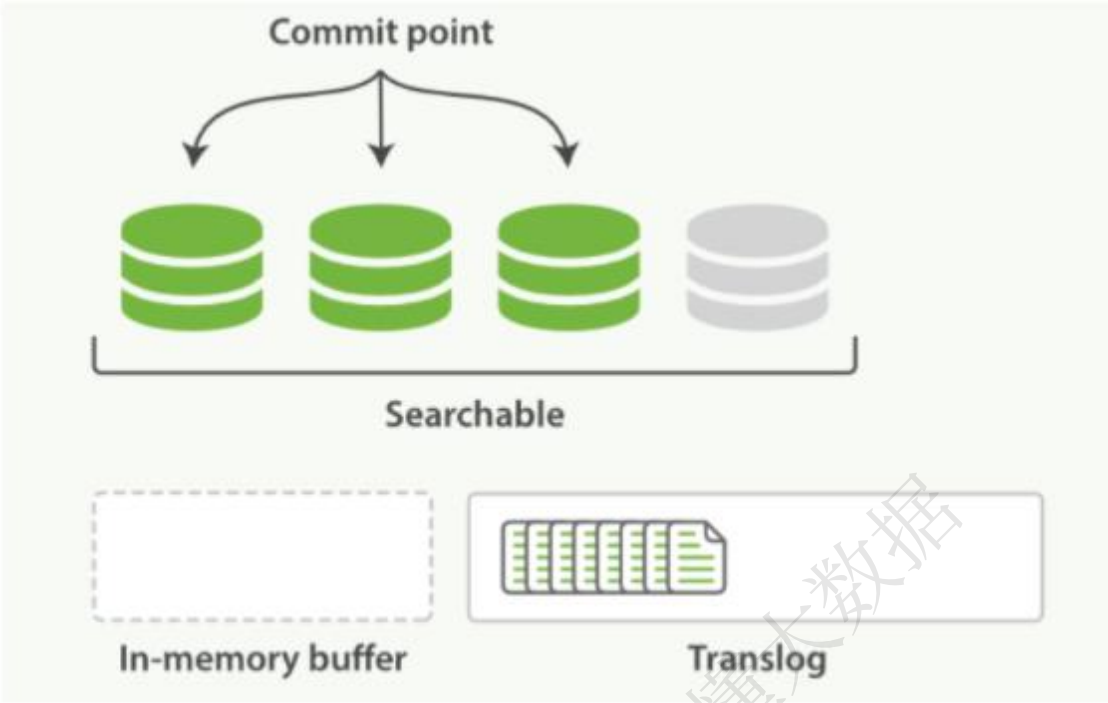


图 2、在执行刷新后清空内存，新文档写入文件系统缓存

不断有新的文档写入，则这一过程将不断重复执行。每隔一秒将生成一个新的 segment，而 translog 文件将越来越大。

文档来自 · 3分钟秒懂大数据

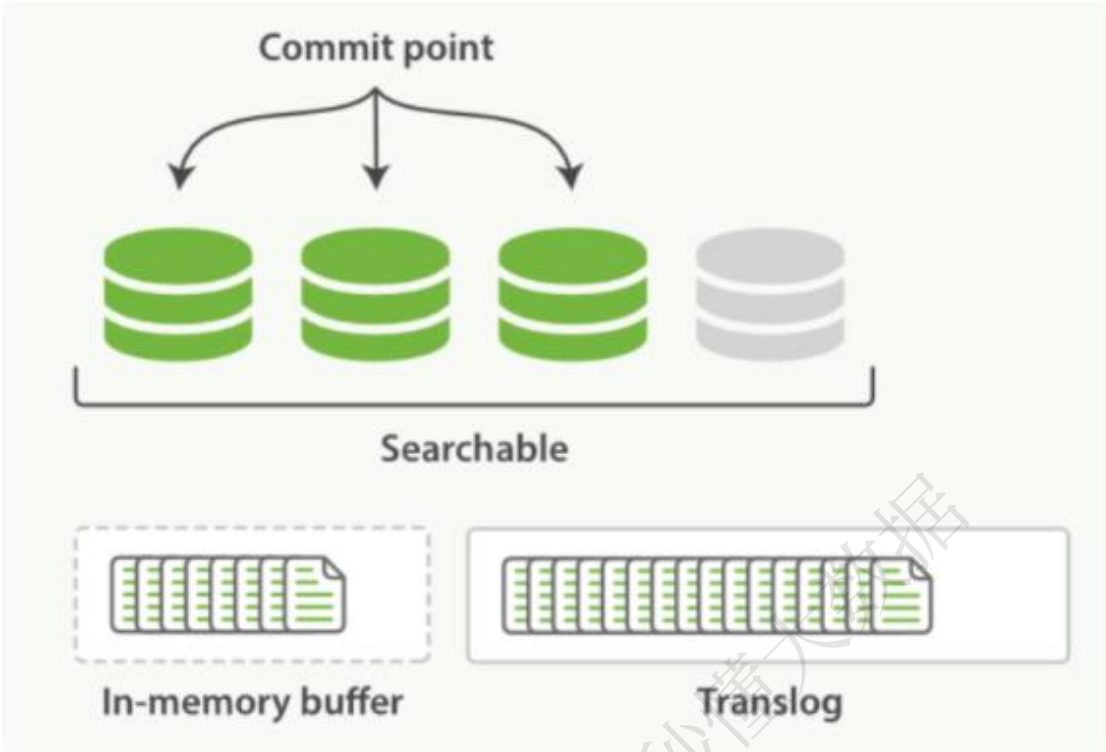


图3、translog不断加入新文档记录

每隔 30 分钟或者 translog 文件变得很大，则执行一次 `fsync` 操作。此时所有在文件系统缓存中的 segment 将被写入磁盘，而 translog 将被删除（此后会生成新的 translog）

文档来自公众号：3分钟秒懂大数据

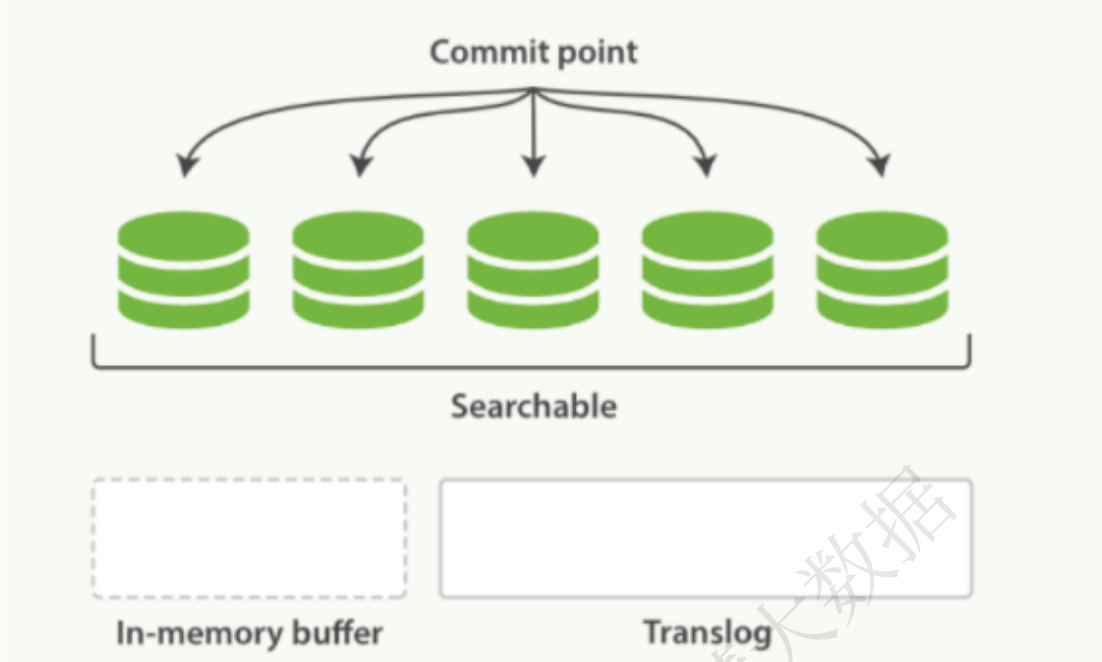


图4、执行fsync后segment写入磁盘，清空内存和translog

由上面的流程可以看出，在两次 `fsync` 操作之间，存储在**内存**和 **文件系统** 缓存中的文档是不安全的，一旦出现断电这些文档就会丢失。**所以 ES 引入了 translog 来记录两次 fsync 之间所有的操作，这样机器从故障中恢复或者重新启动，ES 便可以根据 translog 进行还原**。

当然，**translog 本身也是文件**，存在于内存当中，如果发生断电一样会丢失。因此，`ES` 会在每隔 `5` 秒时间或是一次写入请求完成后将 `translog` 写入磁盘。可以认为一个对文档的操作一旦写入磁盘便是安全的可以复原的，因此只有在当前操作记录被写入磁盘，**ES** 才会将操作成功的结果返回发送此操作请求的客户端。

此外，由于每一秒就会生成一个新的 `segment`，很快将会有大量的 segment。对于一个分片进行查询请求，将会轮流查询分片中的所有 segment，这将降低搜索效率。**因此 ES 会自动启动合并 segment 的工作，将一部分相似大小的 segment 合并成一个新的大 segment**。合并的过程实际上是创建了一个新的 segment，当新 segment 被写入磁盘，所有被合并的旧 segment 被清除。

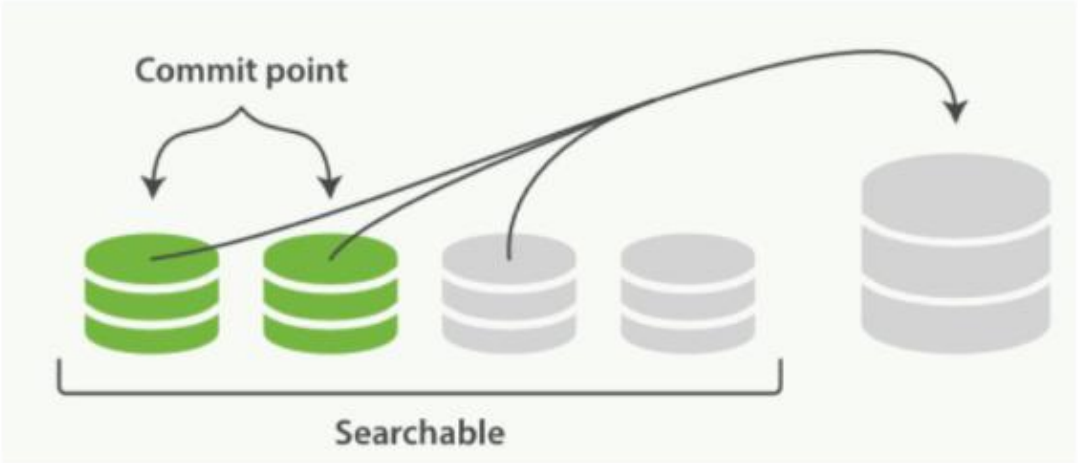


图5、合并segment

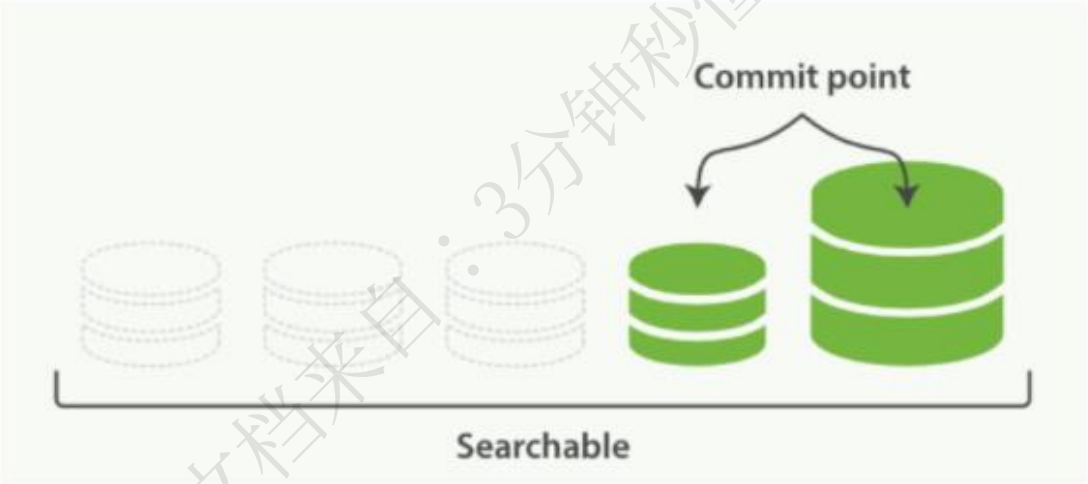


图6、合并完成后删除旧 segment，新 segment 可供搜索

文档来自：3分钟秒懂大数据

5.2 更新（Update）和删除（Delete）文档

ES 的索引是不能修改的，因此**更新**和 **删除** 操作并不是直接在原索引上直接执行。

每一个磁盘上的 segment 都会维护一个 del 文件，用来记录被删除的文件。**每当用户提出一个删除请求，文档并没有被真正删除**，索引也没有发生改变，**而是在 del 文件中标记该文档已被删除**。因此被删除的文档依然可以被检索到，只是在返回

检索结果时被过滤掉了。**每次在启动 segment 合并工作时，那些被标记为删除的文档才会被真正删除**。

更新文档会首先查找原文档，得到该文档的版本号。然后将修改后的文档写入内存，此过程与写入一个新文档相同。同时，旧版本文档被标记为删除，同理，该文档可以被搜索到，只是最终被过滤掉。

读操作（Read）：查询过程

查询的过程大体上分为查询（query）和取回（fetch）两个阶段。这个节点的任务是广播查询请求到所有相关分片，并将它们的响应整合成全局排序后的结果集合，这个结果集合会返回给客户端。

查询阶段

当一个节点接收到一个搜索请求，则这个节点就变成了协调节点。

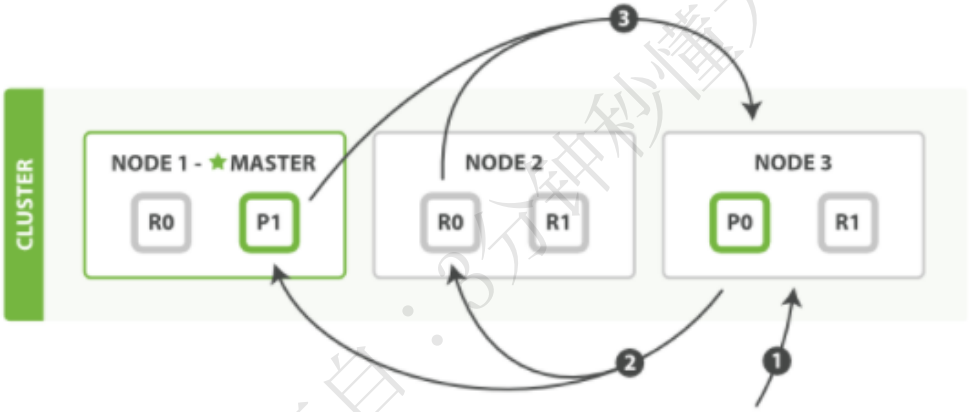


图7、查询过程分布式搜索

第一步是广播请求到索引中每一个节点的分片拷贝。 查询请求可以被某个主分片或某个副本分片处理，**协调节点将在之后的请求中轮询所有的分片拷贝来分摊负载**。

每个分片将会在本地构建一个优先级队列。如果客户端要求返回结果排序中从第 from 名开始的数量为 size 的结果集，则每个节点都需要生成一个 from+size 大小的结果集，因此优先级队列的大小也是 from+size。分片仅会返回一个轻量级的结果给协调节点，包含结果集中的每一个文档的 ID 和进行排序所需要的信息。

协调节点会将所有分片的结果汇总，并进行全局排序，得到最终的查询排序结果。此时查询阶段结束。

取回阶段

查询过程得到的是一个排序结果，标记出哪些文档是符合搜索要求的，此时仍然需要获取这些文档返回客户端。

协调节点会确定实际需要返回的文档，并向含有该文档的分片发送 get 请求；分片获取文档返回给协调节点；协调节点将结果返回给客户端。

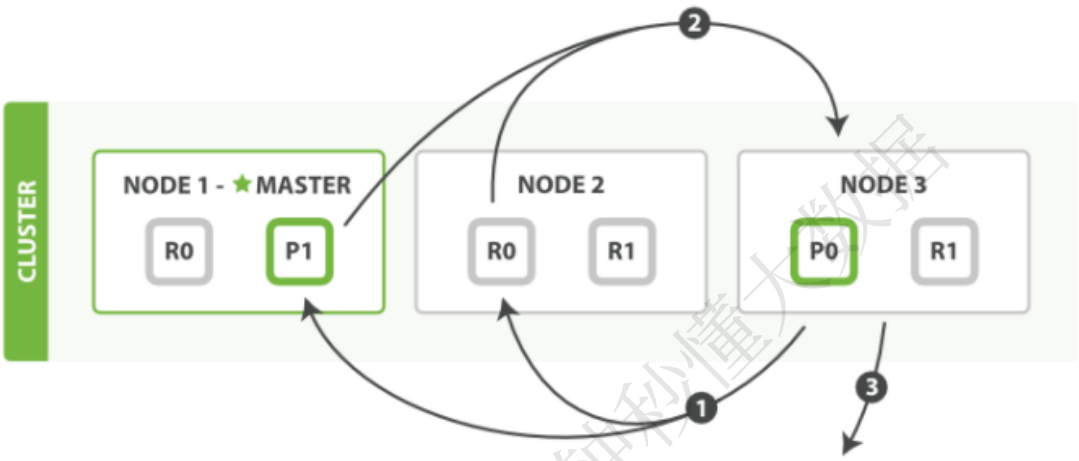


图8、分布式搜索的取回阶段

文档来自：3分钟秒懂大数据