ElasticSearch 6.5.3

# ES核心概念讲解以及命令行操作

## 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.Index（索引）----->6.X-7.X相当于数据库中的表

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。

新建索引库:

PUT index\_icoding

查看刚创建的索引库包含的基本**属性**:

GET index\_icoding

{

"index\_icoding" : {

"aliases" : { },

"mappings" : { },

"settings" : {

"index" : {

"creation\_date" : "1588317754917", #创建时间

"number\_of\_shards" : "5", #分片数（简单来讲就是咱们在ES中所有数据的文件块。）

"number\_of\_replicas" : "1", #副本数也是数据备份数，如果只有一台机器，设置为0

"uuid" : "5NBeBzPBRzq78b5wKrs2Yw", #索引库id

"version" : { #索引库版本号

"created" : "6050399"

},

"provided\_name" : "index\_icoding" #索引库名称

}

}

}

}

分片是什么？

简单来讲就是咱们在ES中所有数据的文件块。

实列场景：

咱们设置了索引A有2个分片，我们向索引A中插入10条数据，那么这10条数据分为5条存储在第一个分片，剩下的5条会存储在另一个分片中。

和主流关系型数据库的表分区的概念有点类似，如果你比较熟悉关系型数据库的话。

注意

1、默认索引是5个分片

2、分片一经设置是不可以修改的，只能新建新索引解决

## 3.Mapping（映射）----->6.X-7.X相当于数据库中的表结构

mapping定义了每个字段的类型、字段所使用的分词器等。相当于关系型数据库中的表结构。

## 4.Document（文档）----->6.X-7.X相当于数据库中的一行一行的记录

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

5.type（类型）

一种type就像一类表。如用户表、角色表等。在Elasticsearch7.X默认type为\_doc

-ES5.X中一个index可以有多种type。

-ES6.X中一个index只能有一种type。

-ES7.X以后，将逐步移除type这个概念，现在的操作已经不再使用，默认\_doc

# ElasticSearch7.4.0集群配置信息

#集群名称

cluster.name: icoding-course

#节点名称

node.name: icoding-1

#是不是有资格主节点

node.master: true

#是否存储数据

node.data: true

#最大集群节点数

node.max\_local\_storage\_nodes: 3

#ip地址

network.host: 0.0.0.0

#端口

http.port: 9201

#内部节点之间沟通端口

transport.tcp.port: 9700

#es7.x 之后新增的配置，节点发现

discovery.seed\_hosts: ["localhost:9700","localhost:9800","localhost:9900"]

#es7.x 之后新增的配置，初始化一个新的集群时需要此配置来选举master

cluster.initial\_master\_nodes: ["icoding-1","icoding-2","icoding-3"]

#数据和存储路径

path.data: /opt/data

path.logs: /opt/logs

# ElasticSearch7.4.0 单机配置信息

cluster.name: icoding-course

node.name: node-1

network.host: 0.0.0.0

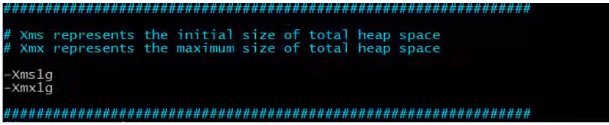
http.port: 9201

cluster.initial\_master\_nodes: ["node-1"]

# 设置ES的JVM占用内存参数

进入/config/jvm.options

默认情况下ES启动JVM最小内存1G，最大内存1G

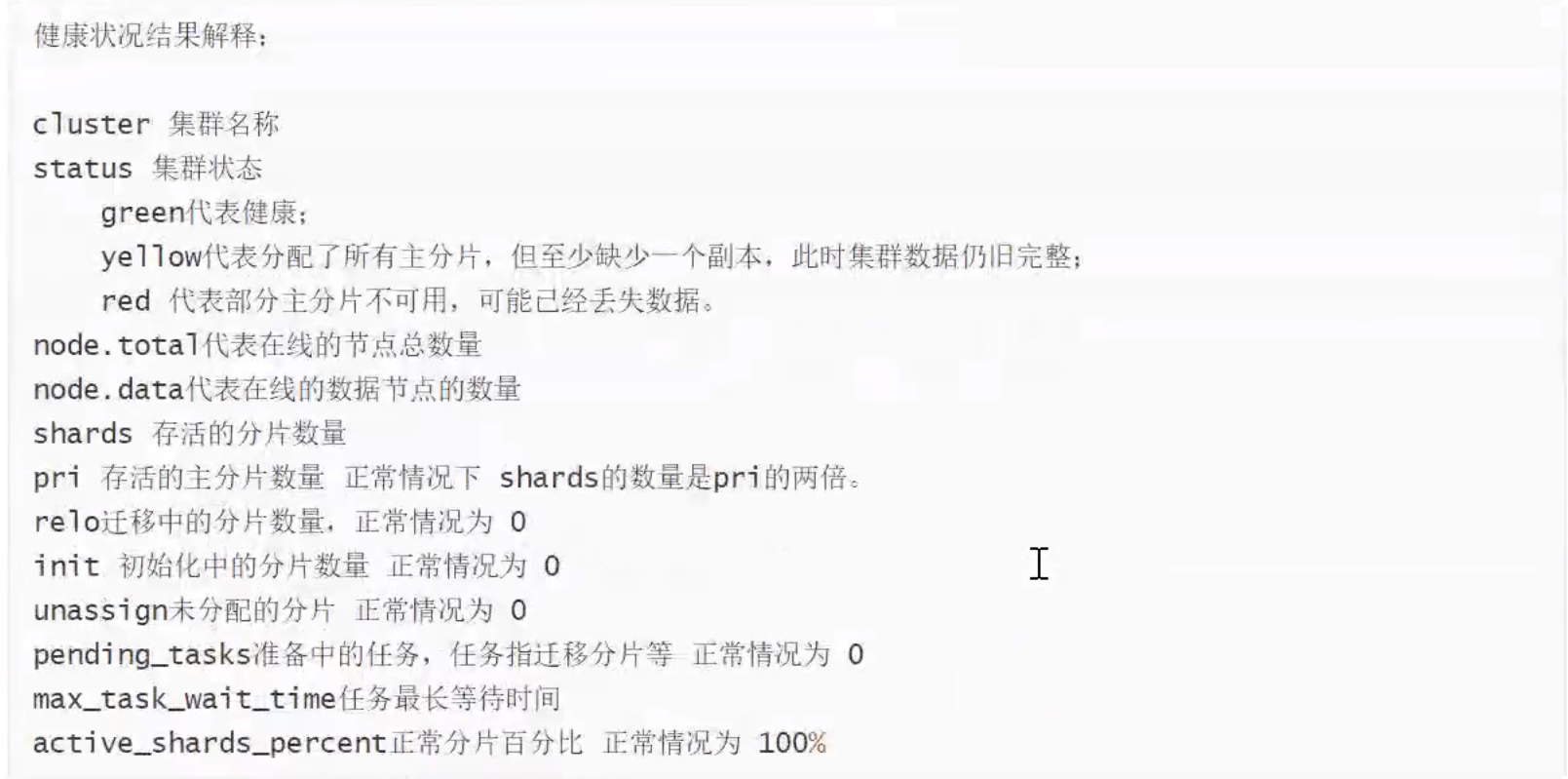




如果是集群修改成256m，这样运行内存就能减少2G左右的占用。

检测集群是否健康,确保9200端口可用,访问链接：

<http://localhost:9200/_cat/health?v>



# Kibana数据可视化分析平台-----配置信息

修改/config/kibana.yml文件

#访问端口

server.port: 5601

# ip地址，0.0.0.0表示可远程访问

server.host: "0.0.0.0"

#kibana服务名

server.name: "kibana-One"

#elasticSearch地址

elasticsearch.hosts:

["http://localhost:9200"]

#请求elasticsearch超时时间，默认为30000，此处可根据情况设置

elasticsearch.requestTimeout: 99999

# Logstash安装使用以及基本配置

## 1.概念

Logstash是一个实时数据收集引擎，可收集各类型数据并对其进行分析，过滤和归纳。按照自己条件分析过滤出符合数据导入到可视化界面。它可以实现多样化的数据源数据全量或增量传输，数据标准格式处理，数据格式化输出等功能，常用于日志处理。工作流程分为三个阶段：

（1）input数据输入阶段，可接受oracle、mysql、postgresql、file、kafka等多种数据源；

（2）filter数据标准格式化阶段，可过滤、格式化数据，如格式化时间、字符串等；

（3）output数据输出阶段，可输出到elasticsearch、mongodb、kafka等接收终端。

那么，接下来将会对每个阶段进行讲解。

## 2.数据采集

**配置数据模板**

E:\elasticsearch-6.5.3\logstash-6.5.3\config目录下新建文件

product\_info\_1.json 内容:

{

"settings": {

"number\_of\_shards": "1"

},

"mappings": {

"doc": {

"properties": {

"product\_info\_id":{

"type": "keyword"

},

"store\_info\_id":{

"type": "keyword"

},

"product\_name":{

"type": "text",

"analyzer": "ik\_max\_word",

"search\_analyzer": "ik\_smart"

},

"spec":{

"type": "text",

"analyzer": "ik\_max\_word",

"search\_analyzer": "ik\_smart"

},

"region\_code":{

"type": "text",

"analyzer": "ik\_max\_word",

"search\_analyzer": "ik\_smart"

},

"price":{

"type": "float"

},

"image\_url":{

"index": false,

"type": "keyword"

},

"create\_time":{

"format":"strict\_date\_optional\_time||epoch\_millis",

"type":"date"

}

}

}

}

}

**配置输入数据源和输出数据源**

E:\elasticsearch-6.5.3\logstash-6.5.3\config目录下新建文件

mysql.conf 内容:

input {

stdin {

}

jdbc {

# 数据库连接

jdbc\_connection\_string => "jdbc:mysql://localhost:3306/my\_test?useUnicode=true&amp;characterEncoding=UTF-8"

# 数据库用户名

jdbc\_user => "root"

# 数据库密码

jdbc\_password => "123456"

# 数据库驱动路径

jdbc\_driver\_library => "E:/m2/repository/mysql/mysql-connector-java/5.1.47/mysql-connector-java-5.1.47.jar"

# 驱动class类

jdbc\_driver\_class => "com.mysql.jdbc.Driver"

# 是否数据解析 需要

jdbc\_paging\_enabled => "true"

# 一次采集多少条数据 50000条

jdbc\_page\_size => "50000"

# 要执行的sql文件

statement => "SELECT \* FROM product\_info\_1 where create\_time > date\_add(:sql\_last\_value,INTERVAL 8 HOUR)"

# 定时配置

schedule => "\* \* \* \* \*"

# 重复运行

record\_last\_run => true

# 记录时间戳的路径

last\_run\_metadata\_path => "E:/elasticsearch-6.5.3/logstash-6.5.3/config/logstash\_metadata"

}

}

output {

stdout {

#日志输出

codec => json\_lines

}

elasticsearch {

#ES的ip地址和端口

hosts => "localhost:9200"

#hosts => ["localhost:9200","localhost:9201","localhost:9202"]

# ES索引库名称

index => "product\_info\_1"

document\_type => "doc"

document\_id => "%{product\_info\_id}"

template => "E:/elasticsearch-6.5.3/logstash-6.5.3/config/product\_info\_1.json"

template\_name => "product\_info\_1"

template\_overwrite => "true"

}

}

**logstash.yml文件配置**

path.data: E:/elasticsearch-6.5.3/logstash-6.5.3/data

path.logs: E:/elasticsearch-6.5.3/logstash-6.5.3/logs

**数据库sql配置**

create database `my\_test`;

use `my\_test`;

DROP TABLE IF EXISTS `product\_info\_1`;

CREATE TABLE `product\_info\_1` (

`product\_info\_id` bigint(20) NOT NULL COMMENT 'id',

`store\_info\_id` bigint(20) DEFAULT NULL COMMENT '所属店铺id',

`product\_name` varchar(100) DEFAULT NULL COMMENT '商品名称',

`spec` varchar(50) DEFAULT NULL COMMENT '规格',

`region\_code` varchar(50) DEFAULT NULL COMMENT '产地',

`price` decimal(10,0) DEFAULT NULL COMMENT '商品价格',

`image\_url` varchar(100) DEFAULT NULL COMMENT '商品图片',

`create\_time` timestamp NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间',

PRIMARY KEY (`product\_info\_id`) USING BTREE,

KEY `FK\_Reference\_1` (`store\_info\_id`) USING BTREE

) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW\_FORMAT=DYNAMIC;

/\*Data for the table `product\_info\_1` \*/

insert into `product\_info\_1`

(`product\_info\_id`,`store\_info\_id`,`product\_name`,`spec`,`region\_code`,`price`,`image\_url`,`create\_time`)

values

(454776435022757888,2,'Java编程思想1','大号','110100','60',NULL,now())

,(454776435081478144,2,'Java编程思想3','大号','110100','60',NULL,now())

,(454776435156975616,2,'Java编程思想5','大号','110100','60',NULL,now())

,(454776435224084480,2,'Java编程思想7','大号','110100','60',NULL,now())

,(454776435291193344,2,'Java编程思想9','大号','110100','60',NULL,now());

## 3.创建索引库

PUT /product\_info\_1

{

"settings": {

"number\_of\_shards": "1"

},

"mappings": {

"doc": {

"properties": {

"product\_info\_id":{

"type": "keyword"

},

"store\_info\_id":{

"type": "keyword"

},

"product\_name":{

"type": "text",

"analyzer": "ik\_max\_word",

"search\_analyzer": "ik\_smart"

},

"spec":{

"type": "text",

"analyzer": "ik\_max\_word",

"search\_analyzer": "ik\_smart"

},

"region\_code":{

"type": "text",

"analyzer": "ik\_max\_word",

"search\_analyzer": "ik\_smart"

},

"price":{

"type": "float"

},

"image\_url":{

"index": false,

"type": "keyword"

},

"create\_time":{

"format":"strict\_date\_optional\_time||epoch\_millis",

"type":"date"

}

}

}

}

}

GET /product\_info\_1

GET /product\_info\_1/\_search

DELETE /product\_info\_1

## 4.运行Logstash

cd E:\elasticsearch-6.5.3\logstash-6.5.3\bin 目录下

输入内容如下:

logstash -f ../config/mysql.conf

# ElasticSearch6.X索引库mapping映射字段详情注解

{

　 "type" : "text", #是数据类型一般文本使用text(可分词进行模糊查询)；keyword无法被分词(不需要执行分词器)，用于精确查找

"analyzer" : "ik\_max\_word", #指定分词器，一般使用最大分词：ik\_max\_word

"normalizer" : "normalizer\_name", #字段标准化规则；如把所有字符转为小写；具体如下举例

"boost" : 1.5, #字段权重；用于查询时评分，关键字段的权重就会高一些，默认都是1；另外查询时可临时指定权重

"coerce" : true, #清理脏数据：1，字符串会被强制转换为整数 2，浮点数被强制转换为整数；默认为true

"copy\_to" : "field\_name", #自定\_all字段；指定某几个字段拼接成自定义；具体如下举例

"doc\_values" : true, #加快排序、聚合操作，但需要额外存储空间；默认true，对于确定不需要排序和聚合的字段可false

"dynamic" : true, #新字段动态添加 true:无限制 false:数据可写入但该字段不保留 'strict':无法写入抛异常

"enabled" : true, #是否会被索引，但都会存储;可以针对一整个\_doc

"fielddata" : false, #针对text字段加快排序和聚合（doc\_values对text无效）；此项官网建议不开启，非常消耗内存

"eager\_global\_ordinals": true, #是否开启全局预加载,加快查询；此参数只支持text和keyword，keyword默认可用，而text需要设置fielddata属性

"format" : "yyyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch\_millis" ,#格式化 此参数代表可接受的时间格式 3种都接受

"ignore\_above" : 100, #指定字段索引和存储的长度最大值，超过最大值的会被忽略

"ignore\_malformed" : false ,#插入文档时是否忽略类型 默认是false 类型不一致无法插入

"index\_options" : "docs" ,

# 4个可选参数

# docs（索引文档号）,

# freqs（文档号 + 词频），

# positions（文档号 + 词频 + 位置，通常用来距离查询），

# offsets（文档号 + 词频 + 位置 + 偏移量，通常被使用在高亮字段）

# 分词字段默认是position，其他的默认是docs

"index" : true, #该字段是否会被索引和可查询 默认true

"fields": {"raw": {"type": "keyword"}} ,#可以对一个字段提供多种索引模式，使用text类型做全文检索，也可使用keyword类型做聚合和排序

"norms" : true, #用于标准化文档，以便查询时计算文档的相关性。建议不开启

"null\_value" : "NULL", #可以让值为null的字段显式的可索引、可搜索

"position\_increment\_gap" : 0 ,#词组查询时可以跨词查询 既可变为分词查询 默认100

"properties" : {}, #嵌套属性，例如该字段是音乐，音乐还有歌词，类型，歌手等属性

"search\_analyzer" : "ik\_max\_word" ,#查询分词器;一般情况和analyzer对应

"similarity" : "BM25",#用于指定文档评分模型，参数有三个：

# BM25 ：ES和Lucene默认的评分模型

# classic ：TF/IDF评分

# boolean：布尔模型评分

"store" : true, #默认情况false,其实并不是真没有存储，\_source字段里会保存一份原始文档。

# 在某些情况下，store参数有意义，比如一个文档里面有title、date和超大的content字段，如果只想获取title和date

"term\_vector" : "no" #默认不存储向量信息，

# 支持参数yes（term存储），

# with\_positions（term + 位置）,

# with\_offsets（term + 偏移量），

# with\_positions\_offsets(term + 位置 + 偏移量)

# 对快速高亮fast vector highlighter能提升性能，但开启又会加大索引体积，不适合大数据量用

}

# Elasticsearch mappings之dynamic的三种状态

一般的，mapping则又可以分为动态映射（dynamic mapping）、静态（显式）映射（explicit mapping）、精确（严格）映射（strict mappings）具体由dynamic属性控制，不同状态所附带属性如下：

* 动态映射（dynamic：true）：动态添加新的字段（或缺省）。
* 静态映射（dynamic：false）：忽略新的字段。在原有的映射基础上，当有新的字段时，不会主动的添加新的映射关系，只作为查询结果出现在查询中。
* 严格模式（dynamic： strict）：如果遇到新的字段，就抛出异常。

一般静态映射用的较多。就像HTML的img标签一样，src为自带的属性，你可以在需要的时候添加id或者class属性。

当然，如果你非常非常了解你的数据，并且未来很长一段时间不会改变，strict不失为一个好选择。

参考文章链接:<https://www.cnblogs.com/Pythonzrq/p/11982645.html>

# 一、索引操作示例

## 创建索引

创建名为mydlq-user的索引与对应Mapping。

方法一、

创建索引的同时把映射也进行添加

PUT /mydlq-user

{

"mappings": {

"doc": {

"dynamic": true,

"properties": {

"name": {

"type": "text",

"fields": {

"keyword": {

"type": "keyword"

}

}

},

"address": {

"type": "text",

"fields": {

"keyword": {

"type": "keyword"

}

}

},

"remark": {

"type": "text",

"fields": {

"keyword": {

"type": "keyword"

}

}

},

"age": {

"type": "integer"

},

"salary": {

"type": "float"

},

"birthDate": {

"type": "date",

"format": "yyyy-MM-dd"

},

"createTime": {

"type": "date"

}

}

}

}

}

方法二、

该方法需要先创建索引再进行mapping映射创建

PUT /mydlq-user/\_mapping

{

"doc": {

"dynamic": true,

"properties": {

"name": {

"type": "text",

"fields": {

"keyword": {

"type": "keyword"

}

}

},

"address": {

"type": "text",

"fields": {

"keyword": {

"type": "keyword"

}

}

},

"remark": {

"type": "text",

"fields": {

"keyword": {

"type": "keyword"

}

}

},

"age": {

"type": "integer"

},

"salary": {

"type": "float"

},

"birthDate": {

"type": "date",

"format": "yyyy-MM-dd"

},

"createTime": {

"type": "date"

}

}

}

}

## 删除索引

删除mydlq-user索引

DELETE /mydlq-user

## 关闭索引

POST mydlq-user/\_close

## 打开索引

POST mydlq-user/\_open

查询映射

GET mydlq-user/\_mapping

添加字段

PUT mydlq-user/\_mapping

{

"properties": {

"sex": {

"type": "text"

}

}

}

# 二、文档操作示例

## 增加文档信息

在索引 mydlq-user 中增加一条文档信息。

POST /mydlq-user/doc

{

"address": "北京市",

"age": 29,

"birthDate": "1990-01-10",

"createTime": 1579530727699,

"name": "张三",

"remark": "来自北京市的张先生",

"salary": 100

}

## 获取文档信息

获取 mydlq-user 的索引 id=1 的文档信息。

GET /mydlq-user/doc/1

## 更新文档信息

更新之前创建的 id=1 的文档信息。

PUT /mydlq-user/doc/1

{

"address": "北京市海淀区",

"age": 29,

"birthDate": "1990-01-10",

"createTime": 1579530727699,

"name": "张三",

"remark": "来自北京市的张先生",

"salary": 100

}

## 删除文档信息

删除之前创建的 id=1 的文档信息。

DELETE /mydlq-user/doc/1

# 三、插入初始化数据

执行查询示例前，先往索引中插入一批数据：

## 1、单条插入

POST mydlq-user/\_doc

{

"name": "零零",

"address": "北京市丰台区",

"remark": "低层员工",

"age": 29,

"salary": 3000,

"birthDate": "1990-11-11",

"createTime": "2019-11-11T08:18:00.000Z"

}

## 2、批量插入

POST \_bulk

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"刘一","address":"北京市丰台区","remark":"低层员工","age":30,"salary":3000,"birthDate":"1989-11-11","createTime":"2019-03-15T08:18:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"陈二","address":"北京市昌平区","remark":"中层员工","age":27,"salary":7900,"birthDate":"1992-01-25","createTime":"2019-11-08T11:15:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"张三","address":"北京市房山区","remark":"中层员工","age":28,"salary":8800,"birthDate":"1991-10-05","createTime":"2019-07-22T13:22:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"李四","address":"北京市大兴区","remark":"高层员工","age":26,"salary":9000,"birthDate":"1993-08-18","createTime":"2019-10-17T15:00:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"王五","address":"北京市密云区","remark":"低层员工","age":31,"salary":4800,"birthDate":"1988-07-20","createTime":"2019-05-29T09:00:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"赵六","address":"北京市通州区","remark":"中层员工","age":32,"salary":6500,"birthDate":"1987-06-02","createTime":"2019-12-10T18:00:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"孙七","address":"北京市朝阳区","remark":"中层员工","age":33,"salary":7000,"birthDate":"1986-04-15","createTime":"2019-06-06T13:00:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"周八","address":"北京市西城区","remark":"低层员工","age":32,"salary":5000,"birthDate":"1987-09-26","createTime":"2019-01-26T14:00:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"吴九","address":"北京市海淀区","remark":"高层员工","age":30,"salary":11000,"birthDate":"1989-11-25","createTime":"2019-09-07T13:34:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"郑十","address":"北京市东城区","remark":"低层员工","age":29,"salary":5000,"birthDate":"1990-12-25","createTime":"2019-03-06T12:08:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"萧十一","address":"北京市平谷区","remark":"低层员工","age":29,"salary":3300,"birthDate":"1990-11-11","createTime":"2019-03-10T08:17:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"曹十二","address":"北京市怀柔区","remark":"中层员工","age":27,"salary":6800,"birthDate":"1992-01-25","createTime":"2019-12-03T11:09:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"吴十三","address":"北京市延庆区","remark":"中层员工","age":25,"salary":7000,"birthDate":"1994-10-05","createTime":"2019-07-27T14:22:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"冯十四","address":"北京市密云区","remark":"低层员工","age":25,"salary":3000,"birthDate":"1994-08-18","createTime":"2019-04-22T15:00:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"蒋十五","address":"北京市通州区","remark":"低层员工","age":31,"salary":2800,"birthDate":"1988-07-20","createTime":"2019-06-13T10:00:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"苗十六","address":"北京市门头沟区","remark":"高层员工","age":32,"salary":11500,"birthDate":"1987-06-02","createTime":"2019-11-11T18:00:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"鲁十七","address":"北京市石景山区","remark":"高员工","age":33,"salary":9500,"birthDate":"1986-04-15","createTime":"2019-06-06T14:00:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"沈十八","address":"北京市朝阳区","remark":"中层员工","age":31,"salary":8300,"birthDate":"1988-09-26","createTime":"2019-09-25T14:00:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"吕十九","address":"北京市西城区","remark":"低层员工","age":31,"salary":4500,"birthDate":"1988-11-25","createTime":"2019-09-22T13:34:00.000Z"}

{"index":{"\_index":"mydlq-user","\_type":"doc"}}

{"name":"丁二十","address":"北京市东城区","remark":"低层员工","age":33,"salary":2100,"birthDate":"1986-12-25","createTime":"2019-03-07T12:08:00.000Z"}

## 3、查询数据

插入完成后再查询数据，查看之前插入的数据是否存在：

GET mydlq-user/\_search 或 GET mydlq-user/\_search?size=10 (注释size为分页查询时每页显示的总条数)

执行后得到下面记录:

{

"took": 2,

"timed\_out": false,

"\_shards": {

"total": 1,

"successful": 1,

"skipped": 0,

"failed": 0

},

"hits": {

"total": 20,

"max\_score": 1,

"hits": [

{

"\_index": "mydlq-user",

"\_type": "\_doc",

"\_id": "BeN0BW8B7BNodGwRFTRj",

"\_score": 1,

"\_source": {

"name": "刘一",

"address": "北京市丰台区",

"remark": "低层员工",

"age": 30,

"salary": 3000,

"birthDate": "1989-11-11",

"createTime": "2019-03-15T08:18:00.000Z"

}

},

{

"\_index": "mydlq-user",

"\_type": "\_doc",

"\_id": "BuN0BW8B7BNodGwRFTRj",

"\_score": 1,

"\_source": {

"name": "陈二",

"address": "北京市昌平区",

"remark": "中层员工",

"age": 27,

"salary": 7900,

"birthDate": "1992-01-25",

"createTime": "2019-11-08T11:15:00.000Z"

}

},

{

"\_index": "mydlq-user",

"\_type": "\_doc",

"\_id": "B-N0BW8B7BNodGwRFTRj",

"\_score": 1,

"\_source": {

"name": "张三",

"address": "北京市房山区",

"remark": "中层员工",

"age": 28,

"salary": 8800,

"birthDate": "1991-10-05",

"createTime": "2019-07-22T13:22:00.000Z"

}

},

{

"\_index": "mydlq-user",

"\_type": "\_doc",

"\_id": "CON0BW8B7BNodGwRFTRj",

"\_score": 1,

"\_source": {

"name": "李四",

"address": "北京市大兴区",

"remark": "高层员工",

"age": 26,

"salary": 9000,

"birthDate": "1993-08-18",

"createTime": "2019-10-17T15:00:00.000Z"

}

},

{

"\_index": "mydlq-user",

"\_type": "\_doc",

"\_id": "CeN0BW8B7BNodGwRFTRj",

"\_score": 1,

"\_source": {

"name": "王五",

"address": "北京市密云区",

"remark": "低层员工",

"age": 31,

"salary": 4800,

"birthDate": "1988-07-20",

"createTime": "2019-05-29T09:00:00.000Z"

}

},

{

"\_index": "mydlq-user",

"\_type": "\_doc",

"\_id": "CuN0BW8B7BNodGwRFTRj",

"\_score": 1,

"\_source": {

"name": "赵六",

"address": "北京市通州区",

"remark": "中层员工",

"age": 32,

"salary": 6500,

"birthDate": "1987-06-02",

"createTime": "2019-12-10T18:00:00.000Z"

}

},

{

"\_index": "mydlq-user",

"\_type": "\_doc",

"\_id": "C-N0BW8B7BNodGwRFTRj",

"\_score": 1,

"\_source": {

"name": "孙七",

"address": "北京市朝阳区",

"remark": "中层员工",

"age": 33,

"salary": 7000,

"birthDate": "1986-04-15",

"createTime": "2019-06-06T13:00:00.000Z"

}

},

{

"\_index": "mydlq-user",

"\_type": "\_doc",

"\_id": "DON0BW8B7BNodGwRFTRj",

"\_score": 1,

"\_source": {

"name": "周八",

"address": "北京市西城区",

"remark": "低层员工",

"age": 32,

"salary": 5000,

"birthDate": "1987-09-26",

"createTime": "2019-01-26T14:00:00.000Z"

}

},

{

"\_index": "mydlq-user",

"\_type": "\_doc",

"\_id": "DeN0BW8B7BNodGwRFTRj",

"\_score": 1,

"\_source": {

"name": "吴九",

"address": "北京市海淀区",

"remark": "高层员工",

"age": 30,

"salary": 11000,

"birthDate": "1989-11-25",

"createTime": "2019-09-07T13:34:00.000Z"

}

},

{

"\_index": "mydlq-user",

"\_type": "\_doc",

"\_id": "DuN0BW8B7BNodGwRFTRj",

"\_score": 1,

"\_source": {

"name": "郑十",

"address": "北京市东城区",

"remark": "低层员工",

"age": 29,

"salary": 5000,

"birthDate": "1990-12-25",

"createTime": "2019-03-06T12:08:00.000Z"

}

}

]

}

}

# 四、查询操作示例

## 1、精确查询(term)

①精确查询

精确查询，查询地址为 北京市通州区 的人员信息：

查询条件不会进行分词，但是查询内容可能会分词，导致查询不到。之前在创建索引时设置 Mapping 中 address 字段存在 keyword 字段是专门用于不分词查询的子字段。

GET mydlq-user/\_search

{

"query": {

"term": {

"address.keyword": {

"value": "北京市通州区"

}

}

}

}

②精确查询-多内容查询

精确查询，查询地址为 北京市丰台区、北京市昌平区 或 北京市大兴区 的人员信息：

GET mydlq-user/\_search

{

"query": {

"terms": {

"address.keyword": [

"北京市丰台区",

"北京市昌平区",

"北京市大兴区"

]

}

}

}

## 2、匹配查询(match)

①匹配查询全部数据与分页

匹配查询符合条件的所有数据，并且设置以 salary 字段升序排序，并设置分页：

GET mydlq-user/\_search

{

"query": {

"match\_all": {}

},

"from": 0,

"size": 10,

"sort": [

{

"salary": {

"order": "asc"

}

}

]

}

②匹配查询数据

匹配查询地址为 通州区 的数据：

GET mydlq-user/\_search

{

"query": {

"match": {

"address": "通州区"

}

}

}

③词语匹配查询

词语匹配进行查询，匹配 address 中为 北京市通州区 的员工信息：

GET mydlq-user/\_search

{

"query": {

"match\_phrase": {

"address": "北京市通州区"

}

}

}

④内容多字段查询

查询在字段 address、remark 中存在 北京 内容的员工信息：

GET mydlq-user/\_search

{

"query": {

"multi\_match": {

"query": "北京",

"fields": ["address","remark"]

}

}

}

## 3、模糊查询(fuzzy)

模糊查询所有以 三 结尾的姓名

GET mydlq-user/\_search

{

"query": {

"fuzzy": {

"name": "三"

}

}

}

## 4、范围查询(range)

查询岁数 ≥ 30 岁的员工数据：

GET /mydlq-user/\_search

{

"query": {

"range": {

"age": {

"gte": 30

}

}

}

}

查询生日距离现在 30 年间的员工数据：

GET mydlq-user/\_search

{

"query": {

"range": {

"birthDate": {

"gte": "now-30y"

}

}

}

}

## 5、通配符查询(wildcard)

查询所有以 “三” 结尾的姓名：

GET mydlq-user/\_search

{

"query": {

"wildcard": {

"name.keyword": {

"value": "\*三"

}

}

}

}

## 6、布尔查询(bool)

查询出生在 1990-1995 年期间，且地址在 北京市昌平区、北京市大兴区、北京市房山区 的员工信息：

GET /mydlq-user/\_search

{

"query": {

"bool": {

"filter": {

"range": {

"birthDate": {

"format": "yyyy",

"gte": 1990,

"lte": 1995

}

}

},

"must": [

{

"terms": {

"address.keyword": [

"北京市昌平区",

"北京市大兴区",

"北京市房山区"

]

}

}

]

}

}

}

# 五、聚合查询操作示例

## 1、Metric 聚合分析

统计员工总数、工资最高值、工资最低值、工资平均工资、工资总和：

GET /mydlq-user/\_search

{

"size": 0,

"aggs": {

"salary\_stats": {

"stats": {

"field": "salary"

}

}

}

}

统计员工工资最低值：

GET /mydlq-user/\_search

{

"size": 0,

"aggs": {

"salary\_min": {

"min": {

"field": "salary"

}

}

}

}

统计员工工资最高值：

GET /mydlq-user/\_search

{

"size": 0,

"aggs": {

"salary\_max": {

"max": {

"field": "salary"

}

}

}

}

统计员工工资平均值：

GET /mydlq-user/\_search

{

"size": 0,

"aggs": {

"salary\_avg": {

"avg": {

"field": "salary"

}

}

}

}

统计员工工资总值：

GET /mydlq-user/\_search

{

"size": 0,

"aggs": {

"salary\_sum": {

"sum": {

"field": "salary"

}

}

}

}

统计员工总数：

GET /mydlq-user/\_search

{

"size": 0,

"aggs": {

"employee\_count": {

"value\_count": {

"field": "salary"

}

}

}

}

统计员工工资百分位：

GET /mydlq-user/\_search

{

"size": 0,

"aggs": {

"salary\_percentiles": {

"percentiles": {

"field": "salary"

}

}

}

}

## 2、Bucket 聚合分析

按岁数进行聚合分桶，统计各个岁数员工的人数：

GET mydlq-user/\_search

{

"size": 0,

"aggs": {

"age\_bucket": {

"terms": {

"field": "age",

"size": "10"

}

}

}

}

按工资范围进行聚合分桶，统计工资在 3000-5000、5000-9000 和 9000 以上的员工信息：

GET mydlq-user/\_search

{

"aggs": {

"salary\_range\_bucket": {

"range": {

"field": "salary",

"ranges": [

{

"key": "低级员工",

"to": 3000

},{

"key": "中级员工",

"from": 5000,

"to": 9000

},{

"key": "高级员工",

"from": 9000

}

]

}

}

}

}

按照时间范围进行分桶，统计 1985-1990 年和 1990-1995 年出生的员工信息：

GET mydlq-user/\_search

{

"size": 10,

"aggs": {

"date\_range\_bucket": {

"date\_range": {

"field": "birthDate",

"format": "yyyy",

"ranges": [

{

"key": "出生日期1985-1990的员工",

"from": "1985",

"to": "1990"

},{

"key": "出生日期1990-1995的员工",

"from": "1990",

"to": "1995"

}

]

}

}

}

}

按工资多少进行聚合分桶，设置统计的最小值为 0，最大值为 12000，区段间隔为 3000：

GET mydlq-user/\_search

{

"size": 0,

"aggs": {

"salary\_histogram": {

"histogram": {

"field": "salary",

"extended\_bounds": {

"min": 0,

"max": 12000

},

"interval": 3000

}

}

}

}

按出生日期进行分桶：

GET mydlq-user/\_search

{

"size": 0,

"aggs": {

"birthday\_histogram": {

"date\_histogram": {

"format": "yyyy",

"field": "birthDate",

"interval": "year"

}

}

}

}

## 3、Metric 与 Bucket 聚合分析

按照员工岁数分桶、然后统计每个岁数员工工资最高值:

GET mydlq-user/\_search

{

"size": 0,

"aggs": {

"salary\_bucket": {

"terms": {

"field": "age",

"size": "10"

},

"aggs": {

"salary\_max\_user": {

"top\_hits": {

"size": 1,

"sort": [

{

"salary": {

"order": "desc"

}

}

]

}

}

}

}

}

}

Elasticsearch-analysis-ik-6.5.3（中文分词插件）

# 一、分词插件安装和测试

analysis-ik插件项目地址：<https://github.com/medcl/elasticsearch-analysis-ik>

需要注意的是es的版本和ik插件的版本必须一致。

## 1. 下载安装ik插件

## 2. 然后在es安装目录的plugins文件下创建一个文件夹analysis-ik（随便命名）

## 3. 解压elasticsearch-analysis-ik-6.5.3.zip文件到analysis-ik目录下

## 4.重启es服务器

## 5.测试IK分词效果查看是否安装成功

postman配置内容如下：

请求方式为：POST

链接地址设置：http://localhost:9202/\_analyze?pretty=true

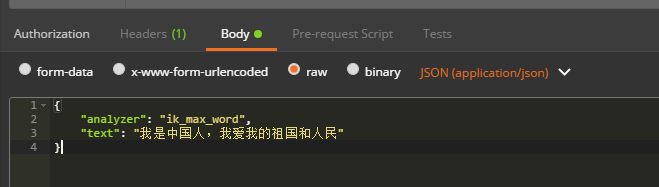
请求Body设置：

{

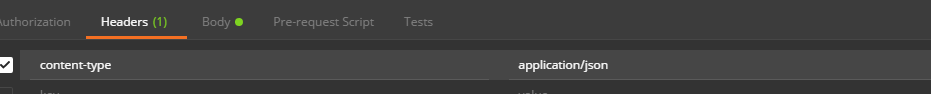
"analyzer": "ik\_max\_word",

"text": "我是中国人，我爱我的祖国和人民"

}



请求头设置：content-type:application/json



请求返回结果：

{

"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": "中国人",

"start\_offset": 2,

"end\_offset": 5,

"type": "CN\_WORD",

"position": 2

},

{

"token": "中国",

"start\_offset": 2,

"end\_offset": 4,

"type": "CN\_WORD",

"position": 3

},

{

"token": "国人",

"start\_offset": 3,

"end\_offset": 5,

"type": "CN\_WORD",

"position": 4

},

{

"token": "我",

"start\_offset": 6,

"end\_offset": 7,

"type": "CN\_CHAR",

"position": 5

},

{

"token": "爱我",

"start\_offset": 7,

"end\_offset": 9,

"type": "CN\_WORD",

"position": 6

},

{

"token": "的",

"start\_offset": 9,

"end\_offset": 10,

"type": "CN\_CHAR",

"position": 7

},

{

"token": "祖国",

"start\_offset": 10,

"end\_offset": 12,

"type": "CN\_WORD",

"position": 8

},

{

"token": "和",

"start\_offset": 12,

"end\_offset": 13,

"type": "CN\_CHAR",

"position": 9

},

{

"token": "人民",

"start\_offset": 13,

"end\_offset": 15,

"type": "CN\_WORD",

"position": 10

}

]

}

## 6.IK中文分词方法简单介绍：

**\*IK中文分词方法有两种分别是ik\_max\_word和ik\_smart,它们的区别如下：**

**ik\_max\_word:** 会将文本做最细粒度的拆分，比如会将“中华人民共和国国歌”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,人,民,共和国,共和,和,国国,国歌”，会穷尽各种可能的组合；

**ik\_smart:** 会做最粗粒度的拆分，比如会将“中华人民共和国国歌”拆分为“中华人民共和国,国歌”。

二、IK中文分词使用

1.给索引mapping映射某个属性创建分词属性

PUT /index\_icoding01

{

"settings": {

"number\_of\_shards": "1"

},

"mappings": {

"doc":{

"dynamic": "strict",

"properties": {

"context": {

"type": "text",

"analyzer": "ik\_max\_word",

"search\_analyzer": "ik\_smart"

},

"grade":{

"type": "integer"

}

}

}

}

}

2.往新建分词索引插入数据

POST /index\_icoding01/doc

{

"context": "中国驻洛杉矶领事馆遭亚裔男子枪击 嫌犯已自首",

"grade": 5

}

POST /index\_icoding01/doc

{

"context": "中韩渔警冲突调查：韩警平均每天扣1艘中国渔船",

"grade": 4

}

..........

POST /index\_icoding01/doc

{

"context": "美国留给伊拉克的是个烂摊子吗",

"grade": 3

}

3.测试分词效果

GET /index\_icoding01/\_search

{

"query":{

"match": {

"context": "中国"

}

}

}

结果:

{

"took" : 1,

"timed\_out" : false,

"\_shards" : {

"total" : 1,

"successful" : 1,

"skipped" : 0,

"failed" : 0

},

"hits" : {

"total" : 4,

"max\_score" : 0.5144432,

"hits" : [

{

"\_index" : "index\_icoding01",

"\_type" : "doc",

"\_id" : "8Y0l0HEBfjM5T20APaSF",

"\_score" : 0.5144432,

"\_source" : {

"context" : "中韩渔警冲突调查：韩警平均每天扣1艘中国渔船",

"grade" : 4

}

},

{

"\_index" : "index\_icoding01",

"\_type" : "doc",

"\_id" : "\_Y0l0HEBfjM5T20AWqS2",

"\_score" : 0.5144432,

"\_source" : {

"context" : "中国驻洛杉矶领事馆遭亚裔男子枪击 嫌犯已自首",

"grade" : 4

}

},

{

"\_index" : "index\_icoding01",

"\_type" : "doc",

"\_id" : "\_o0l0HEBfjM5T20AaKQX",

"\_score" : 0.5144432,

"\_source" : {

"context" : "中国驻洛杉矶领事馆遭亚裔男子枪击 嫌犯已自首",

"grade" : 5

}

},

{

"\_index" : "index\_icoding01",

"\_type" : "doc",

"\_id" : "QY0m0HEBfjM5T20ATqUW",

"\_score" : 0.5144432,

"\_source" : {

"context" : "中国驻洛杉矶领事馆遭亚裔男子枪击 嫌犯已自首",

"grade" : 5

}

}

]

}

}