

# Elasticsearch教程(三)

## 一. 分词器安装

### 1.1 ik分词器

#### 1.1.1 下载

下载地址：<https://github.com/medcl/elasticsearch-analysis-ik/releases>

#### 1.1.2 安装

IK分词器在任何操作系统下安装步骤均一样: 在ES的家目录下的 `plugins` 目录下创建名为 `ik` 的文件夹，然后将下载后的 zip 包拷贝到 `ik` 解压即可

IK分词器提供了两种分词方式：

分词器名称	说明
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”，适合 Phrase 查询
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，适合 Term Query；

#### 1.1.3 验证

使用 `ik_smart` 分词器

```
GET _analyze
{
  "analyzer": "ik_smart",
  "text": ["人终究是孤独的，每个人的人生都有别人参与，却都要自己完成。现在已经学会面对和接受孤独，即使伤疤没磨练的足够硬实，也不再依赖别人给的铠甲"]
}
```

使用 `ik_max_word` 分词器

```
GET _analyze
```

```
{
  "analyzer": "ik_max_word",
  "text": ["人终究是孤独的，每个人的人生都有别人参与，却都要自己完成。现在已经学会面对和接受孤独，即使
  伤疤没磨练的足够硬实，也不再依赖别人给的铠甲"]
}
```

#### 1.1.4 自定义词库

在很多时候，业务上的一些词库极有可能不在IK分词器的词库中，需要去定制属于我们自己的词库。例如下面的例子中，**正井猫**、**up主** 被切分为一个个的字，我们希望这两个词语是不被拆分；另外 **的** 作为中文的停顿词，也不希望出现在分词中，所以我们需要自定义词库和停顿词词库。

```
GET _analyze
```

```
{
  "analyzer": "ik_max_word",
  "text": ["请关注正井猫up主，你们的支持是我肝下去的动力"]
}
```

#### IK分词器的结果

```
1 GET _analyze
2 {
3   "analyzer": "ik_max_word",
4   "text": ["欢迎关注正井猫up主"]
5 }
```

“正井猫”、“up主”被切分为  
一个个字

```
17 {
18   "token": "正",
19   "start_offset": 4,
20   "end_offset": 5,
21   "type": "CN_CHAR",
22   "position": 2
23 },
24 {
25   "token": "井",
26   "start_offset": 5,
27   "end_offset": 6,
28   "type": "CN_CHAR",
29   "position": 3
30 },
31 {
32   "token": "猫",
33   "start_offset": 6,
34   "end_offset": 7,
35   "type": "CN_CHAR",
36   "position": 4
37 },
38 {
39   "token": "up",
40   "start_offset": 7,
41   "end_offset": 9,
42   "type": "ENGLISH",
43   "position": 5
44 },
45 {
46   "token": "主",
47   "start_offset": 9,
48   "end_offset": 10,
49   "type": "CN_CHAR",
50   "position": 6
51 }
```

进入到 `$ES_HOME/plugins/ik/config` 目录下，创建 `custom` 目录，在目录下创建 `mydic.dic`、`ext_stopword.dic` 文件。

在 `mydic.dic` 文件中添加两行内容：

正井猫  
up主

在 `ext_stopword.dic` 中添加一行内容:

的

最后修改 `$ES_HOME/plugins/ik/config/IKAnalyzer.cfg.xml` 文件, 内容如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <comment>IK Analyzer 扩展配置</comment>
  <!--用户可以在这里配置自己的扩展字典 -->
  <entry key="ext_dict">custom/mydic.dic</entry>
  <!--用户可以在这里配置自己的扩展停止词字典-->
  <entry key="ext_stopwords">custom/ext_stopword.dic</entry>
  <!--用户可以在这里配置远程扩展字典 -->
  <!-- <entry key="remote_ext_dict">words_location</entry> -->
  <!--用户可以在这里配置远程扩展停止词字典-->
  <!-- <entry key="remote_ext_stopwords">words_location</entry> -->
</properties>
```

重启 `elasticsearch`, 重新执行如上的命令, 结果如下:

### 自定义词库后再次分词的结果

将"正井猫"、"up主"添加到词库后，这两个词语就不会被切割。

```
1 GET _analyze
2 {
3   "analyzer": "ik_smart",
4   "text": ["关注正井猫up主，你们的支持是我肝下去的动力"]
5 }
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

```
1 {
2   "tokens": [
3     {
4       "token": "关注",
5       "start_offset": 0,
6       "end_offset": 2,
7       "type": "CN_WORD",
8       "position": 0
9     },
10    {
11      "token": "正井猫",
12      "start_offset": 2,
13      "end_offset": 5,
14      "type": "CN_WORD",
15      "position": 1
16    },
17    {
18      "token": "up主",
19      "start_offset": 5,
20      "end_offset": 8,
21      "type": "CN_WORD",
22      "position": 2
23    }
24  ]
25 }
```

## 1.2 pinyin分词器

### 1.2.1 下载

下载地址: <https://github.com/medcl/elasticsearch-analysis-pinyin/releases>

### 1.2.2 安装

pinyin 分词器在任何操作系统下安装步骤均一样: 在ES的家目录下的 `plugins` 目录下创建名为 `pinyin` 的文件夹, 然后将下载后的 zip 包拷贝到 `pinyin` 解压即可

### 1.2.3 验证

执行如下命令:

```
GET _analyze
{
  "analyzer": "pinyin",
  "text": "正井猫"
}
```

#### pinyin分词器的结果

```
1 GET _analyze
2 {
3   "analyzer": "pinyin",
4   "text": "正井猫"
5 }
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

```
1 {
2   "tokens" : [
3     {
4       "token" : "zheng",
5       "start_offset" : 0,
6       "end_offset" : 0,
7       "type" : "word",
8       "position" : 0
9     },
10    {
11      "token" : "zjm",
12      "start_offset" : 0,
13      "end_offset" : 0,
14      "type" : "word",
15      "position" : 0
16    },
17    {
18      "token" : "jing",
19      "start_offset" : 0,
20      "end_offset" : 0,
21      "type" : "word",
22      "position" : 1
23    },
24    {
25      "token" : "mao",
26      "start_offset" : 0,
27      "end_offset" : 0,
28      "type" : "word",
29      "position" : 2
30    }
31  ]
32 }
```

## 1.3 自定义分词器以及应用

对于 `<p>刘德华</p>`, 现在想要得到如下的分词结果:

```
{
  "tokens" : [
    {
```

```

    "token" : "刘德华",
    "start_offset" : 0,
    "end_offset" : 3,
    "type" : "word",
    "position" : 0
  },
  {
    "token" : "liudehua",
    "start_offset" : 0,
    "end_offset" : 3,
    "type" : "word",
    "position" : 0
  },
  {
    "token" : "ldh",
    "start_offset" : 0,
    "end_offset" : 3,
    "type" : "word",
    "position" : 0
  }
]
}

```

### 1.3.1 设置分词器

```

PUT test
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_analyzer": {
          "char_filter": ["html_strip"],
          "tokenizer": "keyword",
          "filter": "my_pinyin_filter"
        }
      },
      "filter": {
        "my_pinyin_filter": {
          "type": "pinyin",
          "keep_first_letter": true,
          "keep_full_pinyin": false,
          "keep_joined_full_pinyin": true,
          "keep_original": true,
          "keep_none_chinese": false,
          "keep_none_chinese_in_joined_full_pinyin": true
        }
      }
    }
  }
}

```

### 1.3.2 验证分词器效果

```
GET test/_analyze
{
  "analyzer": "my_analyzer",
  "text": ["刘德华"]
}
```

#### 自定义分词的执行结果

```
1 PUT test
2 {
3   "settings": {
4     "analysis": {
5       "analyzer": {
6         "my_analyzer": {
7           "char_filter": ["html_strip"],
8           "tokenizer": "keyword",
9           "filter": "my_pinyin_filter"
10        }
11      },
12      "filter": {
13        "my_pinyin_filter": {
14          "type": "pinyin",
15          "keep_first_letter": true,
16          "keep_full_pinyin": false,
17          "keep_joined_full_pinyin": true,
18          "keep_original": true,
19          "keep_none_chinese": false,
20          "keep_none_chinese_in_joined_full_pinyin": true
21        }
22      }
23    }
24  }
25 }
26
27 GET test/_analyze
28 {
29   "analyzer": "my_analyzer",
30   "text": ["刘德华"]
31 }
```

```
1 {
2   "tokens": [
3     {
4       "token": "刘德华",
5       "start_offset": 0,
6       "end_offset": 3,
7       "type": "word",
8       "position": 0
9     },
10    {
11      "token": "liudehua",
12      "start_offset": 0,
13      "end_offset": 3,
14      "type": "word",
15      "position": 0
16    },
17    {
18      "token": "ldh",
19      "start_offset": 0,
20      "end_offset": 3,
21      "type": "word",
22      "position": 0
23    }
24  ]
25 }
```

### 1.3.3 为属性添加分词器

设定 `mappings` 信息，指定索引 `test` 的 `name` 属性的 `analyzer` 自定义的分词器。

```
PUT test/_mapping
{
  "properties": {
    "name": {
      "type": "completion",
      "analyzer": "my_analyzer"
    }
  }
}
```

### 1.3.4 结果验证

执行如下命令添加数据

```
POST test/_bulk
{"index": {}}
{"name": "刘德华"}
{"index": {}}
{"name": "张学友"}
{"index": {}}
{"name": "柳岩"}
```

## 执行前缀建议语句

### 前缀建议案例(一)

```
4 GET test/_search
5 {
6   "suggest": {
7     "name_suggestion": {
8       "prefix": "liu",
9       "completion": {
10        "field": "name"
11      }
12    }
13  }
14 }
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
```

前缀为 liu

拼音以 liu 开始  
的数据都在结果集中

```
19 "name_suggestion" : [
20   {
21     "text" : "liu",
22     "offset" : 0,
23     "length" : 3,
24     "options" : [
25       {
26         "text" : "刘德华",
27         "index" : "test",
28         "type" : "doc",
29         "id" : "w1IrznQB_7iBemPuqVAw",
30         "score" : 1.0,
31         "source" : {
32           "name" : "刘德华"
33         }
34       },
35       {
36         "text" : "柳岩",
37         "index" : "test",
38         "type" : "doc",
39         "id" : "xVlrznQB_7iBemPuqVAw",
40         "score" : 1.0,
41         "source" : {
42           "name" : "柳岩"
43         }
44       }
45     ]
46   }
47 ]
```

### 前缀建议案例(二)

```
13 GET test/_search
14 {
15   "suggest": {
16     "name_suggestion": {
17       "prefix": "liuy",
18       "completion": {
19        "field": "name"
20      }
21    }
22  }
23 }
24
25
26
27
28
29
30
31
32
33
34
35
36
37
```

前缀为 "liuy"

结果集只有 柳岩

```
18 "suggest" : {
19   "name_suggestion" : [
20     {
21       "text" : "liuy",
22       "offset" : 0,
23       "length" : 4,
24       "options" : [
25         {
26           "text" : "柳岩",
27           "index" : "test",
28           "type" : "doc",
29           "id" : "xVlrznQB_7iBemPuqVAw",
30           "score" : 1.0,
31           "source" : {
32             "name" : "柳岩"
33           }
34         }
35       ]
36     }
37   ]
38 }
```

### 前缀建议案例(三)

```
10 GET test/_search
11 {
12   "suggest": {
13     "name_suggestion": {
14       "prefix": "刘",
15       "completion": {
16         "field": "name"
17       }
18     }
19   }
20 }
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
```

输入中文“刘”

“刘”的拼音为“liu”，  
“刘德华”和“柳岩”的拼音  
都是以“liu”开头

```
19 "name_suggestion": [
20   {
21     "text": "刘",
22     "offset": 0,
23     "length": 1,
24     "options": [
25       {
26         "text": "刘德华",
27         "index": "test",
28         "type": "doc",
29         "id": "w1IrznQB_7iBemPuqVAw",
30         "score": 1.0,
31         "source": {
32           "name": "刘德华"
33         }
34       },
35       {
36         "text": "柳岩",
37         "index": "test",
38         "type": "doc",
39         "id": "xvIrznQB_7iBemPuqVAw",
40         "score": 1.0,
41         "source": {
42           "name": "柳岩"
43         }
44       }
45     ]
46   }
47 ]
```

通过如上最后一个结果大家仔细去理解《Elasticsearch教程(一)》中，第5节的开始标红的那句话。

## 二. MySQL数据导入到ES

将MySQL的初始化数据导入到ES的方式可以通过程序的方式和工具的方式。本教程使用 Logstash 来初始化导入。首先将 MySQL 的驱动包拷贝到 `$logstash/core/lib/jars/` 目录下；在 `$logstash/config/` 目录下创建名为 `logstash-mysql-news.conf` 的文件，文件内容如下：

```
input {
  jdbc {
    jdbc_driver_class => "com.mysql.cj.jdbc.Driver"
    jdbc_connection_string => "jdbc:mysql://localhost:3306/es?
useSSL=false&serverTimezone=UTC"
    jdbc_user => root
    jdbc_password => "123456"
    #启用追踪，如果为true，则需要指定tracking_column
    use_column_value => true
    #指定追踪的字段，
    tracking_column => id
    #追踪字段的类型，目前只有数字(numeric)和时间类型(timestamp)，默认是数字类型
    tracking_column_type => "numeric"
    #记录最后一次运行的结果
    record_last_run => true
    #上面运行结果的保存位置
    last_run_metadata_path => "mysql-position.txt"
    statement => "SELECT * FROM news where id > :sql_last_value"
    schedule => "* * * * *"
  }
}

filter {
```



```
mutate {  
  split => { "tags" => ", " }  
}  
}  
output {  
  elasticsearch {  
    document_id => "%{id}"  
    document_type => "_doc"  
    index => "news"  
    hosts => ["http://localhost:9200"]  
  }  
  stdout{  
    codec => rubydebug  
  }  
}
```