

# Elasticsearch教程(一)

## 一. 前序

Elasticsearch是一个基于 **Apache Lucene(TM)** 的开源搜索引擎。无论在开源还是专有领域，Lucene可以被认为是迄今为止最先进、性能最好的、功能最全的搜索引擎库。

但是，Lucene只是一个库。想要使用它，你必须使用Java来作为开发语言并将其直接集成到你的应用中，更糟糕的是，Lucene非常复杂，你需要深入了解检索的相关知识来理解它是如何工作的。

Elasticsearch也使用Java开发并使用Lucene作为其核心来实现所有索引和搜索的功能，但是它的目的是通过简单的 **RESTful API** 来隐藏Lucene的复杂性，从而让全文搜索变得简单。

Elasticsearch的中文网址：<https://www.elastic.co/cn/products/elasticsearch>

### 1.1 正向索引和倒排索引

正向索引与倒排索引，这是在搜索领域中非常重要的两个名词，正向索引通常用于数据库中，在搜索引擎领域使用的最多的就是倒排索引，我们根据如下两个网页来对这两个概念进行阐述：

#### html1

我爱我的祖国，我爱编程

#### html2

我爱编程，我是个快乐的小码农

#### 1.1.1 正向索引

假设我们使用mysql的全文检索，会对如上两句话分别进行分词处理，那么预计得到的结果如下：

我 爱 爱我 祖国 我的祖国 编程 爱编程 我爱编程

我 我爱 爱 编程 爱编程 我爱编程 快乐 码农 小码农

假设我们现在使用正向索引搜索 **编程** 这个词，那么会到第一句话中去查找是否包含有 **编程** 这个关键词，如果有则加入到结果集中；第二句话也是如此。假设现在有成千上百个网页，每个网页非常非常的分词，那么搜索的效率将会非常非常低些。

#### 1.1.2 倒排索引

倒排索引是按照分词与文档进行映射，我们来看看如果按照倒排索引的效果：

关键词	文档名
我	html1,html2,html3
爱	html1,html2
爱我	html1
我爱	html2
祖国	html1
我的祖国	html1
编程	html1,html2
我爱编程	html1,html2
爱编程	html1,html2
快乐	html2
码农	html2
小码农	html2

如果采用倒排索引的方式搜索 `编程` 这个词，那么会直接找到关键词中查找到 `编程`，然后查找到对应的文档，这就是所谓的倒排索引。

## 二. 软件简介以及启动

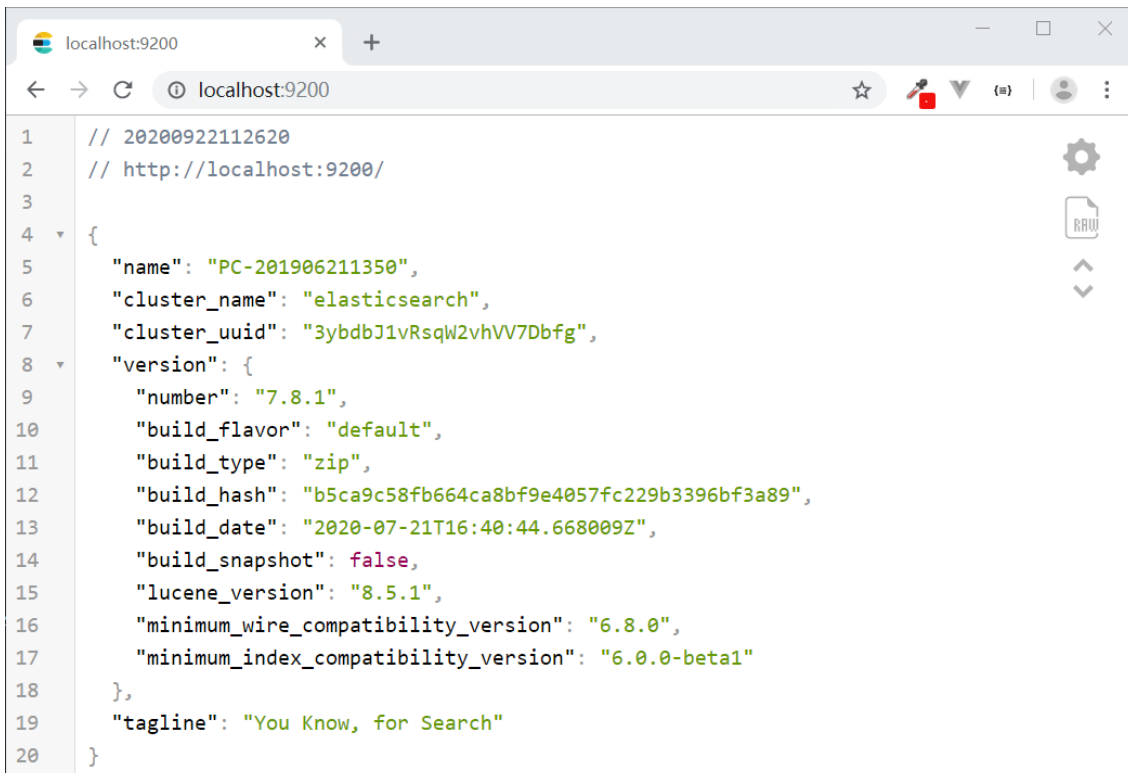
### 2.1 相关软件下载地址

软件名	下载地址
Elasticsearch	<a href="https://www.elastic.co/cn/start">https://www.elastic.co/cn/start</a>
Kibana	<a href="https://www.elastic.co/cn/start">https://www.elastic.co/cn/start</a>
Logstash	<a href="https://www.elastic.co/cn/downloads/logstash">https://www.elastic.co/cn/downloads/logstash</a>

### 2.2 Elasticsearch安装

进入到 `elasticsearch` 解压目录下的 `bin` 目录下，双击 `elasticsearch.bat` 即可启动。

在浏览器地址栏输入: <http://localhost:9200/>，如果出现如下页面表示 `elasticsearch` 启动成功



```
1 // 20200922112620
2 // http://localhost:9200/
3
4 {
5   "name": "PC-201906211350",
6   "cluster_name": "elasticsearch",
7   "cluster_uuid": "3ybdbJ1vRsQW2vhVV7Dbfg",
8   "version": {
9     "number": "7.8.1",
10    "build_flavor": "default",
11    "build_type": "zip",
12    "build_hash": "b5ca9c58fb664ca8bf9e4057fc229b3396bf3a89",
13    "build_date": "2020-07-21T16:40:44.668009Z",
14    "build_snapshot": false,
15    "lucene_version": "8.5.1",
16    "minimum_wire_compatibility_version": "6.8.0",
17    "minimum_index_compatibility_version": "6.0.0-beta1"
18  },
19   "tagline": "You Know, for Search"
20 }
```

## 2.3 Kibana

### 2.3.1 Kibana简介

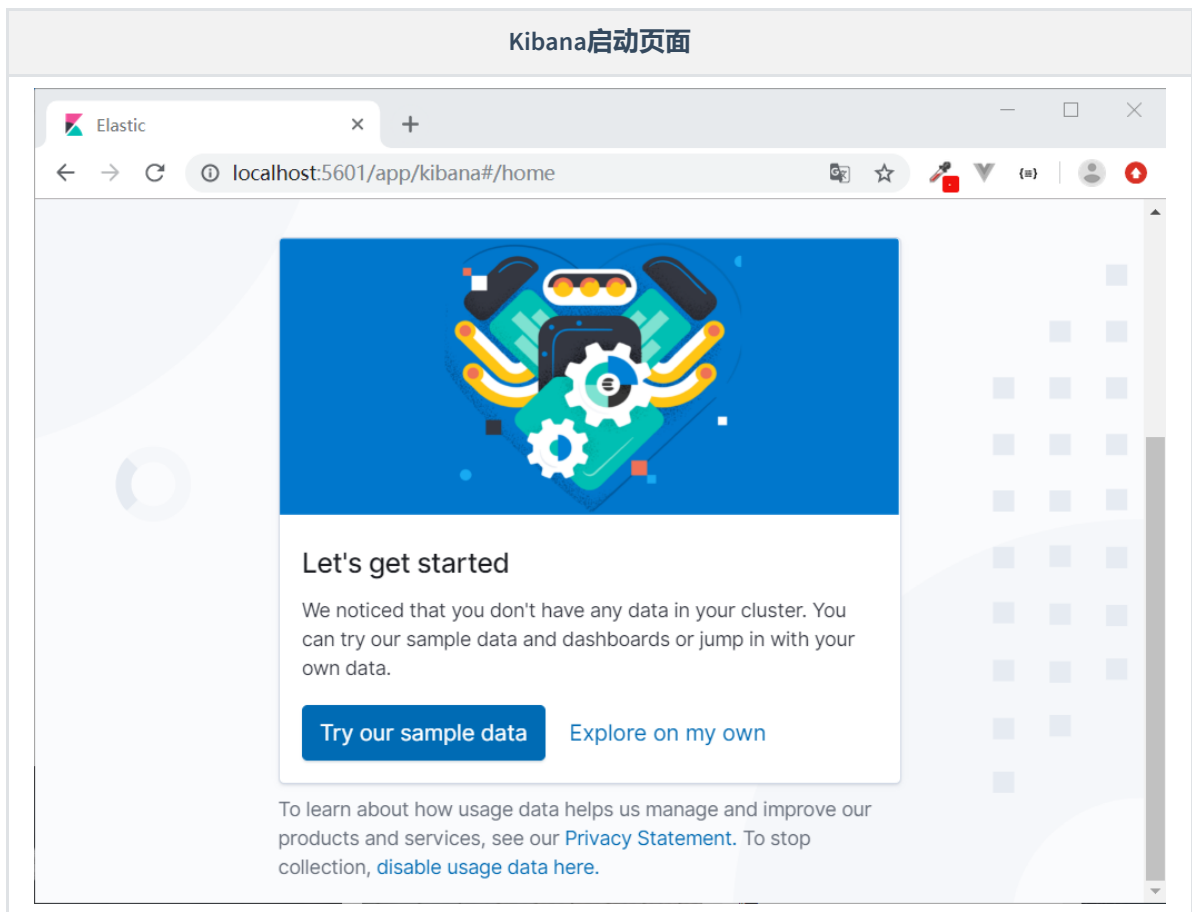
Kibana是世界上最受欢迎的开源日志分析平台ELK Stack中的“K”，它为用户提供了一个工具，用于在存储于Elasticsearch集群中的日志数据进行检索，可视化和构建仪表盘。

Kibana的核心功能是数据查询和分析。使用各种方法，用户可以搜索Elasticsearch中索引的数据，以查找其数据中的特定事件或字符串，以进行根本原因分析和诊断。基于这些查询，用户可以使用Kibana的可视化功能，允许用户使用图表，表格，地理图和其他类型的可视化以各种不同的方式可视化数据。

### 2.3.2 Kibana的启动

进入到 `kibana` 解压目录下的 `bin` 目录下，双击 `kibana.bat` 即可启动 kibana。

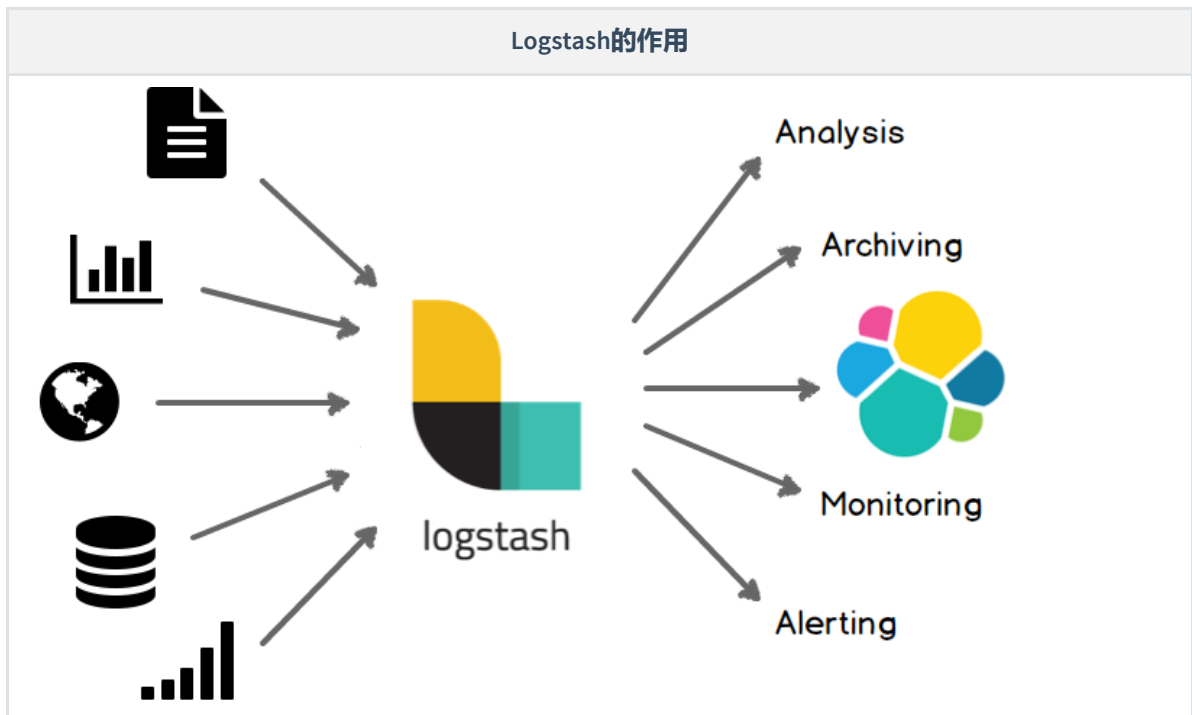
在浏览器地址栏输入：`http://localhost:5601`，出现如下页面代表 kibana 启动成功。



## 2.4 Logstash

### 2.4.1 logstash简介

Logstash是一个开源的服务器端数据处理管道，可以同时从多个数据源获取数据，并对其进行处理，然后将其发送到你最喜欢的“存储”。创建于2009年，于2013年被elasticsearch收购。



## 2.4.2 logstash导入数据

虽然 `kibana` 提供了一些数据集供我们使用，为了加深对 `logstash` 的理解，我们 `movielens` 的电影数据集。

`movielens` 数据集的下载地址为：<http://files.grouplens.org/datasets/movielens/>，进入该网页只需用下载 `ml-latest.zip` 数据即可，如下图所示：



将 `ml-latest.zip` 解压文件中的 `movies.csv` 文件拷贝到 `logstash` 的家目录下；

再将 `logstash` 的 `config` 目录下新建名为 `logstash.conf` 的文件，文件内容如下：

```
input {
  file {
    # 引号的内容为 movies.csv 的实际路径，根据实际情况而定
    path => "D:/logstash-datas/movies.csv"
    start_position => "beginning"
    sincedb_path => "D:/logstash-datas/db_path.log"
  }
}

filter {
  csv {
    separator => ",",
    columns => ["id", "content", "genre"]
  }
}

mutate {
  split => { "genre" => "|" }
  remove_field => ["path", "host", "@timestamp", "message"]
}

mutate {

  split => ["content", "("]
  add_field => { "title" => "%{[content][0]}" }
  add_field => { "year" => "%{[content][1]}" }
```

```

}

mutate {
  convert => {
    "year" => "integer"
  }
  strip => ["title"]
  remove_field => ["path", "host", "@timestamp", "message", "content"]
}

}

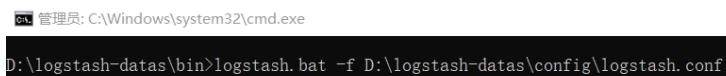
output {
  elasticsearch {
    # 双引号中的内容为ES的地址，视实际情况而定
    hosts => "http://localhost:9200"
    index => "movies"
    document_id => "%{id}"
  }
  stdout {}
}
}

```

打开dos命令行，进入到 **logstash** 的 **bin** 目录下，执行如下命令导入 **movielens** 的数据集。

```
logstash.bat -f D:\logstash-datas\config\logstash.conf
```

### logstash导入数据的命令



```
D:\logstash-datas\bin>logstash.bat -f D:\logstash-datas\config\logstash.conf
```

### 2.4.3 验证

进入到 **kibana** 的命令行页面，执行 **GET \_cat/indices** 验证数据是否成功导入

### 验证 movielens 数据集是否成功导入

History Settings Help									
1	GET	_cat/indices							
1	yellow	open	movies	PugGffIJRniuY8mMuPLP1A	1	1	58099	0	6.8mb
2	green	open	.kibana_task_manager_2	_FFKJDo3T8u2jRxaZi3n_Q	1	0	6	12	68kb
3	green	open	.apm-custom-link	N907EI3T06qtF6GBAs2pQ	1	0	0	0	208b
4	green	open	.kibana_task_manager_1	DoKwr2tuQeuniwd9Wc0Jug	1	0	2	0	38.8kb
5	green	open	kibana_sample_data_ecommerce	18MxsxY7T16nU52Bdlo-Ig	1	0	4675	0	4.5mb
6	green	open	.apm-agent-configuration	20hte1g8R_mdrka5dhm4RQ	1	0	0	0	208b
7	green	open	kibana_sample_data_logs	0KZ8mitEQu2dmbfES8F5CA	1	0	14074	0	10mb
8	yellow	open	2zwjhrjzuo-meow	aP07MiZfS9aC6xg654x42A	1	1	0	0	208b
9	green	open	.kibana_2	jQaRIiTxs-GlbyoMwykS6A	1	0	158	1	1mb
10	green	open	.kibana-event-log-7.8.1-000001	dXbg5cFKSPmxGegCd1uwJw	1	0	1	0	5.2kb
11	green	open	kibana_sample_data_flights	RqX4STE6SH040qGptQ6Wcg	1	0	13059	0	6.1mb
12	green	open	.kibana_1	CHZqOxGzQ60gIy0D8BsIHg	1	0	156	0	1mb
13									

## 三. Elasticsearch的基本概念

3.1 索引

Elasticsearch中的索引有多层的意思：

- a. 某一类文档的集合就构成了一个索引，类比到数据库就是一个数据库(或者数据库表)；
- b. 它还描述了一个动作，就是将某个文档保存在elasticsearch的过程也叫索引；
- c. 倒排索引。

3.2 文档

具体的一条数据，类比到数据库就是一条记录。

ES文档的结构

```
1 {
2   "_index" : "movies",
3   "_type" : "doc",
4   "_id" : "104372",
5   "_version" : 1,
6   "_seq_no" : 21750,
7   "_primary_term" : 1,
8   "found" : true,
9   "source" : {
10    "_year" : 1956,
11    "title" : "Cry in the Night, A",
12    "id" : "104372",
13    "genre" : [
14      "Crime",
15      "Drama",
16      "Film-Noir"
17    ],
18    "@version" : "1"
19  }
20 }
```

索引名

类型

ID

存放具体的数据

3.4 mapping

mapping 是ES每一个文档的约束信息，例如属性的类型，是否能被索引等。

3.5 DSL

DSL 是 ES 的查询语言。

3.6 类比

我们通过大家比较熟悉的 DBMS 与 ES 的基本概念进行类比，加深大家的理解。

DBMS	Elasticsearch
database	Index
table	type(在7.0之后type为固定值_doc)
Row	Document
Column	Field
Schema	Mapping
SQL	DSL(Descriptor Structure Language)

在7.0之前，一个Index可以创建多个类型，从7.0开始，一个索引只能创建一个类型，也就是 `_doc`

## 四. RestAPI

### 4.1 基本CRUD

GET movies/\_search # 查询movies的数据

GET movies/\_count #查询movies的总数

GET \_cat/indices #查看所有的索引

GET movies/\_doc/24 #查询id为24的数据

POST users/\_doc/1 #添加id为1的文档，如果没有指定id，ES会自动生成 { "firstname": "will", "lastname": "smith" }

POST users/\_create/2 #创建id为2的文档，如果索引中已存在相同id，会报错； { "firstname": "will", "lastname": "smith" }

POST users/\_update/2 #在id为2的文档中添加一个age属性，修改结构 { "doc": { "age": 30 } }

DELETE users/\_doc/2 #删除id为2的文档

DELETE users #删除 users 索引

PUT users/\_doc/1 #创建或者修改文档 { "firstname": "Jack", "lastname": "ma" }

PUT users/\_create/2 #创建id为2的文档，如果已存在就报错，如果不存在就创建 { "firstname": "will", "lastname": "smith" }

GET mget #批量查询多个指定的id的数据，也可以批量查询 { "docs": [ { "index": "users", "id": 1 }, { "index": "users", "\_id": 2 } ] }

POST users/ bulk #批量插入数据 { "index": { "id": 3 } } { "firstname": "A", "lastname": "a" } { "index": { "id": 4 } } { "firstname": "B", "lastname": "b" } { "index": { "id": 5 } } { "firstname": "X", "lastname": "x" } { "index": { "\_id": 6 } } { "firstname": "Z", "lastname": "z" }

### 4.2 URI查询

GET movies/\_search?q=2012 #查询所有的属性中只要包含2012的所有数据，泛查询



GET movies/\_search?q=2012&df=title #查询title中包含2012的所有的电影, df(default field)  
或者 GET movies/\_search?q=title:2012

GET movies/\_search?q=title:2012&from=10&size=8 #查询title中包含2012, 从第10条开始, 查询8条数据

GET movies/\_search?q=title:Beautiful Mind #查询title中包含Beautiful或者Mind的所有的数据

GET movies/\_search?q=title:(Beautiful Mind)

GET movies/\_search?q=title:(+Beautiful +Mind)

GET movies/\_search?q=title:"Beautiful Mind" #查询title中包含 "Beautiful Mind" 这个短语的所有的数据

GET movies/\_search?q=title:(Mind AND Beautiful) #查询title中既包含Mind又包含Beautiful的所有的数据, 与顺序没有关系

GET movies/\_search?q=title:(Beautiful NOT Mind) #查询title中包含Beautiful但是不包含mind的所有的数据

GET movies/\_search?q=title:(Beautiful -Mind)

GET movies/\_search?q=title:Beautiful AND year:>=2012 #查询title中包含Beautiful且电影上映时间在2012年之后的所有的数据

GET movies/\_search?q=year:>=2018 #查询2018年之后上映的电影

GET movies/\_search?q=year:(>=2012 AND <2018) #查询在2012到2017年上映的电影

GET movies/\_search?q=year:{2015 TO 2017} #查询2016年到2017年上映的电影, 必须以 ] 结尾

GET movies/\_search?q=title:Min?x # ? 代表一个字母

GET movies/\_search?q=title:Min\* # 查询title中包含以 Min开头的字母的电影

## 五. Analysis

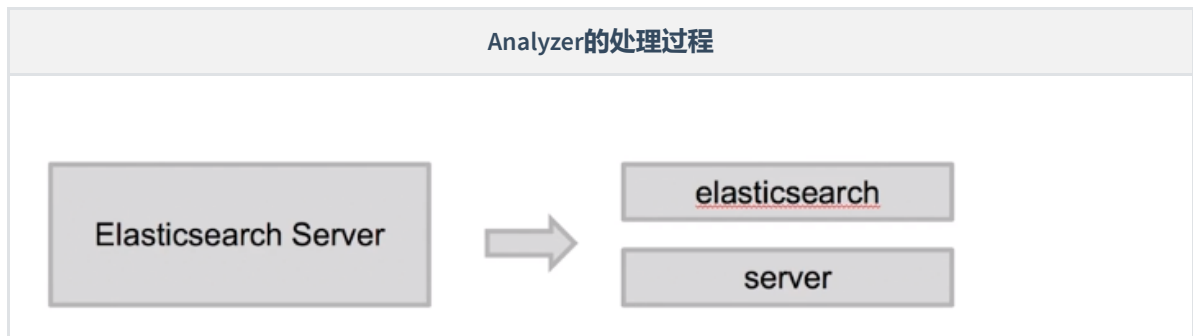
analysis(只是一个概念), 文本分析是将全文本转换为一系列单词的过程, 也叫分词。analysis是通过analyzer(分词器)来实现的, 可以使用Elasticsearch内置的分词器, 也可以自己去定制一些分词器。除了在数据写入的时候进行分词处理, 那么在查询的时候也可以使用分析器对查询语句进行分词。

analyzer是由三部分组成, 例如有

Hello a World, the world is beautifu

:

1. Character Filter: 将文本中html标签剔除掉。
2. Tokenizer: 按照规则进行分词，在英文中按照空格分词。
3. Token Filter: 去掉stop word(停顿词，a, an, the, is, are等)，然后转换小写



## 5.1 内置分词器

分词器名称	处理过程
Standard Analyzer	默认的分词器，按词切分，小写处理
Simple Analyzer	按照非字母切分(符号被过滤)，小写处理
Stop Analyzer	小写处理，停用词过滤(the, a, this)
Whitespace Analyzer	按照空格切分，不转小写
Keyword Analyzer	不分词，直接将输入当做输出
Pattern Analyzer	正则表达式，默认是\W+(非字符串分隔)

## 5.2 内置分词器示例

### 5.2.1 Standard Analyzer

```
GET _analyze
{
  "analyzer": "standard",
  "text": "2 Running quick brown-foxes leap over lazy dog in the summer evening"
}
```

### 5.2.2 Simple Analyzer

```
GET _analyze
{
  "analyzer": "simple",
  "text": "2 Running quick brown-foxes leap over lazy dog in the summer evening"
}
```

### 5.2.3 Stop Analyzer

```
GET _analyze
{
  "analyzer": "stop",
  "text": "2 Running quick brown-foxes leap over lazy dog in the summer evening"
}
```

#### 5.2.4 Whitespace Analyzer

```
GET _analyze
{
  "analyzer": "whitespace",
  "text": "2 Running quick brown-foxes leap over lazy dog in the summer evening"
}
```

#### 5.2.5 Keyword Analyzer

```
GET _analyze
{
  "analyzer": "keyword",
  "text": "2 Running quick brown-foxes leap over lazy dog in the summer evening"
}
```

#### 5.2.6 Pattern Analyzer

```
GET _analyze
{
  "analyzer": "pattern",
  "text": "2 Running quick brown-foxes leap over lazy dog in the summer evening"
}
```