elasticsearch

2018年5月10日 13:32

基础概念

集群和节点

集群是由多个节点构成的

Clumter_name 集群名字

每个节点都是根据集群名字来加入集群的

索引

含有相同属性的文档集合

类型

索引可以定义一个或多个类型，翁当必须属于一个类型

文档

文档是可以被索引的基本数据单位

例如：索引相当于sql里的database

文档相当于sql里的一个表

文档相当于sql里的一条数据

与索引相关的分片与备份

分片 每个索引都有多个分片，每个分片是一个Lucene索引

好处：假设一个索引的数据量很大，就会造成硬盘的压力也很大，同时搜索速度也会出现瓶颈，那么就可以分为多个分片，从而分担压力，分片同时还允许用户水平扩展和拆分，以及分布式的操作，可以提高索引以及其他操作的效率

备份 拷贝一份分片就完成了分片的备份

好处：当一个主分片失败或者出现问题时，备份的分片就可以代替工作，从而提高了es的可用性，备份的分片还可以执行搜索操作，以分担搜索的压力

ES默认会创建5个分片，一个备份，这个数量是可以修改的

分片的数量只能在创建索引时指定，后期无法修改

备份可以动态修改

elasticsearch搭建

2018年5月14日 23:51

Jdk1.8

head插件需要环境

Node.js(内含npm)

`grunt` 也必须安装

我使用的是5.5.2 需要

ROOT用户无法启动 必须创建用户并给用户操作文件夹的权限才能启动

启动

进入${ES_HOME}

./bin/elasticsearch 直接启动

或者

./bin/elasticsearch -d 后台启动

yml配置文件

Master

```
cluster.name: xiaobo
node.name: slave1
#0.0.0.0允许外网访问
network.host: 0.0.0.0
http.port: 8200

#bootstrap的错误
bootstrap.memory_lock: false
bootstrap.system_call_filter: false

discovery.zen.ping.unicast.hosts: ["0.0.0.0"]

#解决跨域问题
http.cors.enabled: true
http.cors.allow-origin: "*"
```

节点

```
cluster.name: xiaobo
node.name: slave2

network.host: 0.0.0.0
http.port: 8201

bootstrap.memory_lock: false
bootstrap.system_call_filter: false

discovery.zen.ping.unicast.hosts: ["192.168.218.128:9300"]

http.cors.enabled: true
http.cors.allow-origin: "*"
```

Elasticsearch使用

2018年5月15日 0:04

- ## RESTFul API

  基本格式 http:<ip>:<port>/<索引>/<类型>/<文档id>

- ## 常用HTTP动词

  GET/PUT/POST/DELETE

- ## 创建索引

  非结构化创建

  结构化创建

- ## 添加索引

  ### 1.在head插件直接创建

  不太方便，json格式需要手动校验

  head插件

  PUT book/novel/_mappings

  {"novel":{"properties":{"title":{"type":"text"}}}}

  ### 2.postman测试

  PUT /perple

```
{"settings":{
"number_of_shards":3,
"number_of_replicas":1


},
"mappings":{
        "man":{
                "properties":{
                        "name":{
                                "type":"text"
                        },
                        "country":{
                                "type":"keyword"
                        },
                        "age":{
                                "type":"integer"
                        },
                        "date":{
                                "type":"date",
                                "format":"yyyy-MM-dd || yyyy-MM-dd HH:mm:ss || epach_millis"
                        }
```

```
                }
            },
            "woman":{}
        }
        }
```

- 插入数据

1.指定id

```
PUT /perple/man/1

{
    "name":"小波",
    "country":"CHINA",
    "age":25,
    "date":"1993-07-15"
}
```

2.自动生成id

```
POST /perple/man

{
    "name":"斌哥",
    "country":"CHINA",
    "age":18,
    "date":"2000-07-15"
}
```

- 修改数据
  - 直接修改文档

```
POST /perple/man/1/_update

{
    "doc":{"name":"谁是小波"}
}
```

  - 通过脚本修改

```
POST /perple/man/1/_update

{
    "script":{
        "lang":"painless",
        "inline":"ctx._source.age += 3"//ctx  es下文参数
    }
}
```

或

```
{
    "script":{
        "lang":"painless",
        "inline":"ctx._source.age = params.age",
        "params":{
            "age":100
        }
    }
```

}

- 删除
  - 删除文档

    DELETE /perple/man/1
  - 删除索引

    1.直接在head插件中删除，动作-删除-输入确认

    2.DELETE /perple
- 查询
  - 简单查询

    指定id查询

    ```
    GET /book/novel/1
    ```

    查询所有

    ```
    POST /book/_search
    {
        "query":{
            "match_all":{}
        }
    }
    ```

    分页查询

    ```
    POST /book/_search
    {
        "query":{
            "match_all":{}
        },
        "from":0,
        "size":5
    }
    ```
  - 条件查询

    普通条件查询

    ```
    POST /book/_search
    {
        "query":{
            "match":{
                "title":"ElasticSearch"
            }
        }
    }
    ```

    自定义排序

    ```
    POST /book/_search
    {
        "query":{
            "match":{
                "title":"ElasticSearch"
            }
        },
        "sort":[
            {"publish_date":{"order":"desc"}}  //按出版日期倒序  默认正序
        ]
    ```

```
}
```

- 聚合查询

  单个聚合条件

```
POST /book/_search
{
    "aggs":{
        "group_by_word_count":{
            "terms":{
                "field":"word_count"
            }
        }
    }
}
```

  多个聚合条件

```
POST /book/_search
{
    "aggs":{
        "group_by_word_count":{
            "terms":{
                "field":"word_count"
            }
        },
        "group_by_publish_date":{
            "terms":{
                "field":"publish_date"
            }
        }
    }
}
```

  功能函数

  统计

```
POST /book/_search
    {
        "aggs":{
            "grades_word_count":{
                "stats":{
                    "field":"word_count"
                }
            }
        }
    }
```

  响应

```
{  ...,
   "aggregations": {
     "grades_word_count": {
       "count": 11,
       "min": 1000,
       "max": 5000,
       "avg": 2090.909090909091,
       "sum": 23000
     }
```

```
  }
}
```

指定函数名

```
POST /book/_search
        {
                "aggs":{
                        "grades_word_count":{
                                "min":{
                                        "field":"word_count"
                                }
                        }
                }
        }
```

响应

```
{...,
 "aggregations": {
      "grades_word_count": {
        "value": 1000
      }
   }
}
```

Elasticsearch高级查询

2018年5月15日 10:58

- 子条件查询 特定字段查询所指特定值
  - QuryContext

    在查询过程中，除了判断文档是否满足查询条件外，ES还会计算一个_sorce 来标识匹配的程度，旨在判断目标文档和查询条件匹配的有多好。

    常用查询
    - 全文本查询 针对文本类型数据
      - *模糊匹配*

        POST /book/_search

```
{
    "query":{
        "match":{
            "author":"小波"
        }
    }
}
```

      - *习语匹配*

```
模糊查询会匹配查询条件的多个词语
比如
{
    "query":{
        "match":{
            "title":"ElasticSearch入门"
        }
    }
}
返回数据既有关于ElasticSearch的 也有关于入门的
```

      - POST /book/_search

```
{
    "query":{
        "match_phrase":{
            "title":"ElasticSearch入门"
        }
    }
}
```

        把查询条件当作一个整体去查询，条件中ElasticSearch入门是一个词

      - *多个字段匹配*
      - POST /book/_search

```
{
    "query":{
        "multi_match":{
            "query":"ElasticSearch",
            "fields":["title","author"]
        }
    }
}
```

- 语法查询
- POST /book/_search

```
{
    "query":{
        "query_string":{
            "query":"(ElasticSearch AND 入门) OR Java OR python"
        }
    }
}
```

- 指定字段
- POST /book/_search

```
{
    "query":{
        "query_string":{
            "query":"ElasticSearch OR python",
            "fields":["title","author"]
        }
    }
}
```

- 字段级别查询 针对结构化数据，如数字、日期等
  - POST /book/_search 字数为1000的书籍

```
{
    "query":{
        "term":{
            "word_count":1000
        }
    }
}
```

  - 范围查询
  - POST /book/_search 字数在1000-2000的书籍

```
{
    "query":{
        "range":{
            "word_count":{
                "gte":1000,
                "lte":2000
            }
        }
    }
}
```

- Filter context

在查询过程中，只判断该文档是否满足条件，只有YES活着NO。

POST /book/_search

```
{
    "query":{
        "bool":{
            "filter":{
                "term":{
                    "word_count":1000
                }
            }
        }
    }
}
```

- 复合条件查询 以一定的逻辑组合子条件查询
  - 固定分数查询

```
POST /_search  //分数都是1
{
    "query":{
        "constant_score":{
            "filter":{
                "match":{
                    "title":"elasticsearch"
                }
            }
        }
    }
}
```

指定分数

```
POST /_search  //分数都是1
{
    "query":{
        "constant_score":{
            "filter":{
                "match":{
                    "title":"elasticsearch"
                }
            },
            "boost":2
        }
    }
}
```

  - 布尔查询
    - 相当于 OR
      - POST /_search

```
{
    "query":{
        "bool":{
            "should":[
                {
                    "match":{
                        "title":"elasticsearch"
                    }
                },
                {
                    "match":{
                        "author":"小波"
                    }
                }
            ]
        }
    }
}
```

    - 相当于 AND

```
{
    "query":{
        "bool":{
            "must":[
                {
                    "match":{
                        "title":"elasticsearch"
                    }
                },
```

```
                {
                    "match":{
                        "author":"斌哥"
                    }
                }
            ]
        }
    }
}
```

- AND 之后过滤

```
{
    "query":{
        "bool":{
            "must":[
                {
                    "match":{
                        "title":"elasticsearch"
                    }
                },
                {
                    "match":{
                        "author":"斌哥"
                    }
                }
            ],
            "filter":[
                {
                    "term":{
                        "word_count":3000
                    }
                }
            ]
        }
    }
}
```

- NOT 关键词must_not

ES安装中文分词器

2018年5月15日 22:38

/usr/local/java/elasticsearch-5.5.2/bin/elasticsearch-plugin install https://github.com/medcl/elasticsearch-analysis-ik/releases/download/v5.5.2/elasticsearch-analysis-ik-5.5.2.zip

配置拼音分词器

(1)创建索引时就指定settings

PUT 192.168.218.128:9200/book

```
{
  "index": {
    "analysis": {
      "analyzer": {
        "ik_pinyin_analyzer": {
          "type": "custom",
          "tokenizer": "ik_smart",
          "filter": ["my_pinyin", "word_delimiter"]
        }
      },
      "filter": {
        "my_pinyin": {
          "type": "pinyin",
          "first_letter": "prefix",
          "padding_char": " "
        }
      }
    }
  }
}
```

【分片和备份会自己设置】

(2)mapping中指定需要使用分词的字段

PUT user/friend/_mapping?pretty

```
{
  "friend": {
    "properties": {
      "userId": {
        "type": "text"
      },
      "friendId": {
        "type": "text"
      },
      "friendHeadimg": {
        "type": "text"
      },
      "friendName": {
        "type": "text",
        "fields": {
          "pinyin": {
            "type": "text",
```

```
          "store": "no",
          "term_vector": "with_positions_offsets",
          "analyzer": "ik_pinyin_analyzer",
          "boost": 10
        }
       }
      },
      "friendPhone": {
       "type": "integer"
      }
    }
  }
}
```

DDBES测试数据

2018年5月23日 10:16

创建索引

PUT /user

```
{
  "index": {
    "analysis": {
      "analyzer": {
        "ik_pinyin_analyzer": {
          "type": "custom",
          "tokenizer": "ik_smart",
          "filter": [
            "my_pinyin",
            "word_delimiter"
          ]
        },
        "email_url_analyzer": {
          "type": "custom",
          "tokenizer": "uax_url_email",
          "filter": [
            "trim"
          ]
        },
        "index_email_analyzer": {
          "type": "custom",
          "tokenizer": "standard",
          "filter": [
            "lowercase",
            "name_ngram_filter",
            "trim"
          ]
        },
        "search_email_analyzer": {
          "type": "custom",
          "tokenizer": "standard",
          "filter": [
            "lowercase",
            "trim"
          ]
        }
      },
      "char_filter": {
        "digit_only": {
          "type": "pattern_replace",
          "pattern": "\\D+",
          "replacement": ""
        }
      },
      "tokenizer": {
        "digit_edge_ngram_tokenizer": {
          "type": "edgeNGram",
          "min_gram": "1",
          "max_gram": "15",
          "token_chars": [
            "digit"
          ]
        }
      },
```

Mappings设置

PUT /user/friend/_mapping?pretty

```
{
  "friend": {
    "properties": {
      "userId": {
        "type": "text"
      },
      "friendId": {
        "type": "text"
      },
      "friendHeadimg": {
        "type": "text"
      },
      "friendName": {
        "type": "text",
        "fields": {
          "pinyin": {
            "type": "text",
            "store": "no",
            "term_vector": "with_positions_offsets",
            "analyzer": "ik_pinyin_analyzer",
            "boost": 10
          }
        }
      },
      "friendPhone": {
        "type": "text",
        "analyzer": "index_email_analyzer",
        "search_analyzer": "search_email_analyzer"
      }
    }
  }
}
```

```
    "filter": {
      "my_pinyin": {
        "type": "pinyin",
        "first_letter": "prefix",
        "padding_char": " "
      },
      "name_ngram_filter": {
        "type": "ngram",
        "min_gram": "1",
        "max_gram": "20"
      }
    }
  }
 }
}
```