Elastic索引

索引操作

创建索引 文档

- 基本语法 `PUT twitter`
- 索引名称限制：
    - 只能小写
    - 不能包括 `\`，`/`，`*`，`?`，`"`，`<`，`>`，`|`，空格，`,`，`#`
    - 不能包括冒号，7.0之前可以用，7.0之后不可以
    - 不能使用 + – 开头

- 不能包括 . 或者 ..
- 不能超过 255 个字节

JSON | 复制代码

```
PUT heshen_test_v1
{

  "settings": {
    "index":{
      "number_of_replicas": 2, //副本数2
      "number_of_shards": 3,//分片3
      "max_result_window": 1000000, //最大返回数据 1000000
      "write" : {
          "wait_for_active_shards" : "2" //等待多少分片执行完成
        }
    }
  },
  "mappings": { //映射字段
    "_doc":{ // 约定都使用_doc ,可以不实用_doc，但是不能使用_开头，
              //7.0之后创建的时候，不用加 _doc
       "properties":{
          "itemName":{
             "type" : "text"
          },
          "weight":{
             "type" : "scaled_float",
             "scaling_factor": 100
          },
          "price":{
             "type" : "long"
          }
       }
    }
  },
  "aliases": {//别名，用处很多
    "heshen_test": {}
  }
}
```

获取索引

- `GET heshen_test` 获取指定索引信息
- `GET *` 获取所有的索引信息

- `GET _all` 获取所有的索引信息
- `GET es_erp_purchase_*` 获取通配符匹配的索引信息

删除索引

- `DELETE heshen_test_v1`
- 删除索引必须是索引名，不能通过别名删除索引

是否存在

- `HEAD heshen_test` 索引是否存在

打开/关闭索引

- `POST heshen_test_v1/_close`
- `POST heshen_test_v1/_open`
- 被关闭的索引禁止读写,只能展示元信息

索引统计

- `GET` `/_stats`
- `GET` `/heshen_test_v1/_stats`

索引别名

文档

创建修改别名

- 带有过滤器的别名提供了创建相同索引的不同“视图”的简单方法。过滤器可以使用查询DSL定义，并应用于所有搜索、计数、按查询删除以及类似于此别名的操作。
- 在创建别名时可以指定路由值。
- 如果一个别名只映射了一个真实索引，则可以使用别名进行index api(即索引文档，写文档)，但如果一个别名同一时间映射了多个索引，默认是不能直接使用别名进行索引文档，因为ES不知道文档该发往哪个索引。可以使用is_write_index属性为一个别名下的其中一个索引指定为写索引，此时则可以直接使用别名进行index api的调用。

JSON | 复制代码

```
POST _aliases
{
  "actions": [
    {
      "add": {
        "index": "heshen_test_v1",
        "alias": "heshen",
        "is_write_index": true,
        "routing": "1",
        "filter": {
          "term": {
            "remarks": "测试"
          }
        }
      }
    },
    {
      "add": {
        "index": "heshen_test_v2",
        "alias": "heshen"
      }
    }
  ]
}
```

删除别名

JSON 复制代码

```
POST _aliases
{
  "actions": [
    {
      "remove": {
        "index": "heshen_test_v2",
        "alias": "heshen"
      }
    },
    {
      "remove": {
        "index": "heshen_test_v1",
        "alias": "heshen"
      }
    }
  ]
}
```

Reindex

文档

- es的索引一旦创建，分片，以及字段类型都不允许修改，所以一般只能创建新的索引，然后把数据导入
- reindex支持把数据从一个索引拷贝到另一个索引中
- reindex不会复制索引的设置，包括mapping ,settings

普通复制

JSON 复制代码

```
POST _reindex
{
  "source": {
    "index": "heshen_test_v1"
  },
  "dest": {
    "index": "heshen_test_v2"
  }
}
```

其他参数

- 在dest中 添加 version_type (见下文类型介绍) 字段可以实现根据版本进行复制
- 默认情况下，version冲突会终止reindex进程，可以通过 conflicts 属性来设置不结束，只统计冲突数量
- 在dest中 op_type 可以设置操作类型，可以设置只复制不存在的数据
- 可以通过 query属性来设置那些数据被复制
- source的index 可以是一个列表 ， 来将多个索引的数据复制到一个索引
- source的 _source 可以设置只复制部分字段
- 可以用过 script 来修改文档

JSON | 复制代码

```
POST _reindex
{
  "size": 12,//只处理12条数据
  "conflicts": "proceed",//冲突计数
  "source": {
    "index": "heshen_test_v1", //也可以是  "index":[ "heshen_test_v1","heshen_test_v3"]
    "query": { //设置源数据那些数据需要复制
      "term": {
        "weight": {
          "value": 12
        }
      }
    },
   "_source": ["weight", "itemName"],//只复制部分字段
   "sort": { "date": "desc" },//设置处理数据排序
   "script": {
      "inline": "if (ctx._source.foo == 'bar') {ctx._version++; ctx._source.remove('foo')}",
      "lang": "painless"
   }
  },
  "dest": {
    "index": "heshen_test_v2",
    "version_type": "external", //版本复制
    "op_type":"create" //只会创建不存在的数据
  }
}
```

索引模板

索引模板是什么

- 索引模板是可以在创建一个新的索引的时候，自动应用的模板，包括设置和映射。
- 模板只在创建索引的时候使用，更改模板不会对现有索引产生影响
- 在创建的时候，自定义的设置，映射优先于 模板当中的。

创建索引模板

JSON 复制代码

```
PUT _template/template_heshen
{
  "index_patterns": [//匹配模式
    "heshen*"
  ],
  "order" : 0,//优先级
  "settings": {//预定的设置
    "index": {
      "number_of_replicas": 2,
      "number_of_shards": 3,
      "max_result_window": 1000000
    }
  },
  "mappings": {//预定的mappings
    "_doc": {
       "properties": {
         "created_name": {
           "type": "keyword"
         }
       }

    },
    "version": 123
  }
}
```

删除索引模板

- `DELETE / _template/template_heshen`

获取索引模板

- `GET /_template/template_heshen`

模板是否存在

- `HEAD /_template/template_heshen`

模板匹配

- 多个模板可以匹配一个索引，多个相同的配置根据order字段来确定使用那个

Mapping操作

更新Mapping

- 映射已经创建后，一般不可以更改,除了更改子对象的字段，或者 ignore_above 属性
- 可以新增一个字段映射

JSON 复制代码

```
PUT heshen_test_v1/_mapping/_doc
{
  "properties":{
     "img":{
       "type":"keyword"
     }
  }
}
```

- 也可以批量添加字段映射

JSON 复制代码

```
PUT /twitter-1,twitter-2/_mapping/_doc
{
  "properties": {
    "user_name": {
      "type": "text"
    }
  }
}
```

获取Mapping

- `GET /heshen_test_v1/_mapping` 获取整个Mapping信息
- `GET /heshen_test_v1/_mapping/_doc`
- `GET /heshen_test_v1/_mapping/field/img` 获取某个字段的映射

Settings操作

更新Settings 文档

- 注意分片数不可以修改

```JSON
PUT /heshen_test_v1/_settings
{
    "index" : {
        "number_of_replicas" : 1
    }
}
```

获取Settings

- `GET heshen_test_v1/_settings`

文档 Document

插入

指定id插入，如果存在则是更新

```JSON
PUT heshen_test_v1/_doc/2
{
  "itemName": "C货2",
  "weight": 12,
  "price":20
}
```

es自动生成id （使用post代替put）

```
POST heshen_test_v1/_doc/2
{...}
```

指定操作类型,指定类型为创建，如果id已经存在，就会返回失败

```
PUT heshen_test_v1/_doc/4/_create
{...}

PUT heshen_test_v1/_doc/4?op_type=create
{...}

```

超时时间

```
PUT heshen_test_v1/_doc/4?timeout=10ms
{...}
```

版本号-乐观锁更新

- 默认es采用内部版本控制，每一次更新版本号+1，我们可以使用version_type=external 来启用外部版本号功能。
- 请求的版本号需要是非负正整数
- 默认es的内部版本控制为一定要与当前版本号相同才能更新成功
- （启用之后，之后好像就不用在每次带version_type）

JSON | 复制代码

```
PUT heshen_test_v1/_doc/2?version=17
{...}
当文档version=17的时候保存成功

PUT heshen_test_v1/_doc/2?version=17&version_type=external_gte
{...}
当文档version<=17的时候成功

PUT heshen_test_v1/_doc/2?version=17&version_type=external
{...}
当文档version<17的时候成功
```

version_type类型

internal： 完全相同才成功

external_gt || external :当请求的版本号大于的时候才成功

external_gte： 当请求的版本号大于等于的时候才成功

自动创建索引。

如果索引不存在，自动创建索引，并且应用索引模板，字段类型es会动态生成。可以通过action.auto_create_index 来控制。

```
PUT _cluster/settings
{
    "persistent": {
        "action.auto_create_index": "heshen*"//只允许某些索引创建
    }
}

PUT _cluster/settings
{
    "persistent": {
        "action.auto_create_index": "false" //全部不允许
    }
}

PUT _cluster/settings
{
    "persistent": {
        "action.auto_create_index": "true" //全部允许
    }
}
```

更新

- 对于并发的数据冲突，es采用乐观锁的方式来解决数据冲突，对于一些以最新数据为准的数据，可以采用自定义版本号的方式解决数据写入冲突。默认还可以设置乐观锁重试次数。

基于id部分字段更新

- 存在的字段会被更新
- 不存在的字段会添加，映射不存在的会自动添加映射

```
POST heshen_test_v1/_doc/1/_update
{
   "doc": {
     "itemName":"C货112"
   }
}
```

基于脚本更新

- 基于es的script脚本进行更新

JSON | 复制代码

```
POST heshen_test_v1/_doc/1/_update
{
  "script": {
     "source":"ctx._source.remarks='和天下'"
  }
}

POST heshen_test_v1/_doc/1/_update
{
  "script": {
    "source": "ctx._source.weight=params.count",
    "lang": "painless",
    "params": {
      "count": 4
    }
  }
}
```

Upsert

- 普通更新如果文档不存在，会报错
- 而加了upsert之后，如果文档不存在，会初始化文档

JSON | 复制代码

```
POST heshen_test_v1/_doc/6/_update
{
  "script": {
    "source": "ctx._source.weight=params.count",
    "lang": "painless",
    "params": {
      "count": 4
    }
  },
  "upsert": {
    "itemName": "C货2",
    "weight": 12,
    "price": 20
  }
}
```

更新基于条件

JSON | 复制代码

```
POST heshen_test_v1/_update_by_query
{
    "script": {
    "source": "ctx._source.remarks=ctx._source.remarks+'*'",
    "lang": "painless"
  },
  "query": {
    "terms": {
      "_id": [
        "1"
      ]
    }
  }
}
```

如何使用脚本

链接

脚本构成

```JSON
  "script": {
    "lang":   "...",  //脚本使用的语言 默认为 painless
    "source" | "id": "...", //
    "params": { ... } //参数
  }
```

搜索返回字段

- es在第一次执行脚本的时候会把他缓存起来，所以如果脚本里有参数的话，一般是要使用参数字段，而不是硬编码

```JSON
GET heshen_test_v1/_search
{
  "script_fields": {
    "weight_big": {
      "script": {
        "lang":   "expression",
        "source":"doc['weight'] * sp",
        "params": {
           "sp": 100
        }
      }
    }
  }
}
```

操作数据

- 在reindex 和 update_by_query 可以设置数据的属性，甚至删除数据

JSON 复制代码

```
noop: 设置 ctx.op = “noop” 。如果你的脚本并没有对原来的doc做任何更改。这将导致 reindex 忽略该doc。这将在响应的 noop 中被展示。

delete: 设置ctx.op = “delete”，如果你的脚本如此设定，target index中的该doc会被被删除。这将在响应的 deleted 中被展示。
```

删除

基于id进行删除

JSON 复制代码

```
DELETE heshen_test_v1/_doc/1
```

基于条件删除

JSON 复制代码

```
POST heshen_test_v1/_delete_by_query
{
  "query": {
    "terms": {
      "_id": [
        "1"
      ]
    }
  }
}
```

Bulk 批量增删改

- bulk是es提供的一种批量增删改

语法

除delete操作意外，所有的操作必须是 `一对JSON` ，而且每个JSON `不能换行` ， `相邻JSON必须换行` ，

- 格式：

  ```
  POST _bulk
  { action: { metadata }}
  { request body        }
  { action: { metadata }}
  { request body        }
  ```

操作类型

- `create` 如果文档不存在就创建，但如果文档存在就返回错误
- `index` 如果文档不存在就创建，如果文档存在就更新
- `update` 更新一个文档，如果文档不存在就返回错误
- `delete` 删除一个文档，如果要删除的文档id不存在，就返回错误

示例

JSON | 复制代码

```
POST _bulk
{"index":{"_index":"heshen_test_v1","_type":"_doc","_id":"4"}}
{"_doc":{"itemName":"C货3","weight":12,"price":20}}

POST heshen_test_v1/_doc/_bulk
{"delete": {"_id": 5}}
```

ps:在es的bulk操作中，可以把索引，type加在操作前面，这样在json中可以不写出 index ,type