Elastic Search: Document APIs[1, 2]

CRUD: Create, Read, Update, and Delete

Weimao Ke

Drexel University

Table of contents

Single Document APIs

Index API

Adds or updates a document to an index:

```
PUT  /<index>/_doc/<_id>

POST  /<index>/_doc/

PUT  /<index>/_create/<_id>

POST  /<index>/_create/<_id>
```

- _*doc* will update an existing document
- _*create* will index it IF it does not exist yet
- An ID will be auto-generated if not specified

Index API

Example:

```
POST twitter/_doc/
{
    "user" : "kimchy",
    "post_date" : "2009-11-15T14:12:12",
    "message" : "trying out Elasticsearch"
}
```

Get API

Retrieves a specific document from an index:

```
GET <index>/_doc/<_id>

HEAD <index>/_doc/<_id>

GET <index>/_source/<_id>

HEAD <index>/_source/<_id>
```

- *GET* to retrieve the document;
- *HEAD* to verify that the document exists;

Get API

Example:

```
GET twitter/_doc/0
```

```
{
    "_index" : "twitter",
    "_type" : "_doc",
    "_id" : "0",
    "_version" : 1,
    "_seq_no" : 10,
    "_primary_term" : 1,
    "found": true,
    "_source" : {
        "user" : "kimchy",
        "date" : "2009-11-15T14:12:12",
        "likes": 0,
        "message" : "trying out Elasticsearch"
    }
}
```

Get API

Example:

```
HEAD twitter/_doc/0
```

Elasticsearch returns a status code:

```
200 – OK
```

if the document exists.

Or:

```
404 – Not Found
```

if it doesn’t.

Delete API

Removes a document from an index.

```
DELETE /<index>/_doc/<_id>
```

Delete API

Example:

```
DELETE /twitter/_doc/1
```

```
{
    "_shards" : {
        "total" : 2,
        "failed" : 0,
        "successful" : 2
    },
    "_index" : "twitter",
    "_type" : "_doc",
    "_id" : "1",
    "_version" : 2,
    "_primary_term": 1,
    "_seq_no": 5,
    "result": "deleted"
}
```

Delete API

Removes documents that match a query.

```
POST /twitter/_delete_by_query
{
  "query": {
    "match": {
      "message": "some message"
    }
  }
}
```

Delete API

Response to a delete by query request:

```
{
  "took" : 147,
  "timed_out": false,
  "total": 119,
  "deleted": 119,
  "batches": 1,
  "version_conflicts": 0,
  "noops": 0,
  "retries": {
    "bulk": 0,
    "search": 0
  },
  "throttled_millis": 0,
  "requests_per_second": -1.0,
  "throttled_until_millis": 0,
  "failures" : [ ]
}
```

Delete API

Delete all tweets from the twitter index:

```
POST twitter/_delete_by_query?conflicts=proceed
{
  "query": {
    "match_all": {}
  }
}
```

Delete documents from multiple indices:

```
POST /twitter,blog/_delete_by_query
{
  "query": {
    "match_all": {}
  }
}
```

Update API

Updates a document using a (automatic) script.

```
POST /<index>/_update/<_id>
```

Update API

Example:

```
  PUT test/_doc/1
  {
      "counter" : 1,
      "tags" : ["red"]
  }
```

Now increments the counter:

```
POST test/_update/1
{
    "script" : {
        "source": "ctx._source.counter += params.count",
        "lang": "painless",
        "params" : {
            "count" : 4
        }
    }
}
```

Update API

You can also add a tag to the list of tags:

```
POST test/_update/1
{
    "script" : {
        "source": "ctx._source.tags.add(params.tag)",
        "lang": "painless",
        "params" : {
            "tag" : "blue"
        }
    }
}
```

Update API

Update by query:

```
POST twitter/_update_by_query
{
  "script": {
    "source": "ctx._source.likes++",
    "lang": "painless"
  },
  "query": {
    "term": {
      "user": "kimchy"
    }
  }
}
```

Multi-document APIs

Multi get (mget) API

Retrieves multiple documents by ID.

GET /_mget
GET /<index>/_mget

Example:

```
GET /twitter/_mget
{
    "ids" : ["1", "2"]
}
```

Multi get (mget) API

Example:

```
GET /twitter/_doc/_mget
{
    "docs" : [
        {
            "_id" : "1"
        },
        {
            "_id" : "2"
        }
    ]
}
```

Multi get (mget) API

Example:

```
GET /_mget
{
    "docs" : [
        {
            "_index" : "twitter",
            "_id" : "1"
        },
        {
            "_index" : "twitter",
            "_id" : "2"
        }
    ]
}
```

Bulk API

Performs multiple indexing or delete or update operations in a single API class.

POST /_bulk

POST /<index>/_bulk

Example:

```
POST _bulk
{ "index" : { "_index" : "test", "_id" : "1" } }
{ "field1" : "value1" }
{ "delete" : { "_index" : "test", "_id" : "2" } }
{ "create" : { "_index" : "test", "_id" : "3" } }
{ "field1" : "value3" }
{ "update" : {"_id" : "1", "_index" : "test"} }
{ "doc" : {"field2" : "value2"} }
```

Reindex API

Copies documents from one index to another.

`POST /_reindex`

Example:

```
POST _reindex
{
  "source": {
    "index": "twitter"
  },
  "dest": {
    "index": "new_twitter"
  }
}
```

Reindex API

Reindex select documents with a query:

```
POST _reindex
{
  "source": {
    "index": "twitter",
    "query": {
      "term": {
        "user": "kimchy"
      }
    }
  },
  "dest": {
    "index": "new_twitter"
  }
}
```

Term Vectors APIs

Term Vectors

Retrieves information and statistics for terms in the fields of a particular document.

GET `/twitter/_termvectors/1`

Example, term vectors for document #1:

```
GET /twitter/_termvectors/1
```

Term vectors based on the "message" field of document #1:

```
GET /twitter/_termvectors/1?fields=message
```

Term Vectors

Term Vetors API parameters:

```
GET /twitter/_termvectors/1
{
  "fields" : ["text"],
  "offsets" : true,
  "payloads" : true,
  "positions" : true,
  "term_statistics" : true,
  "field_statistics" : true
}
```

- *term_statistics*: true to return total term frequency (ttf) and document frequency (DF);
- *field_statistics*: true to return document count, sum of doc frequencies, and sum of total term frequencies in this field;

Term Vectors

Dynamically generate term vectors:

```
GET /twitter/_termvectors
{
  "doc" : {
    "fullname" : "John Doe",
    "text" : "twitter test test test"
  }
}
```

Term Vectors

With a different field analyzer:

```
GET /articles/_termvectors
{
    "doc": {
      "abstract": "information science principles"
    },
    "term_statistics" : true,
    "field_statistics" : false,
    "positions": false,
    "offsets": false,
    "per_field_analyzer" : {
      "abstract": "standard"
    }
}
```

Term Vectors

Term filtering:

```
GET /imdb/_termvectors
{
    "doc": {
      "plot": "When wealthy industrialist Tony Stark
          is forced to build an armored suit after a
          life-threatening incident, he ultimately ..."
    },
    "term_statistics" : true,
    "field_statistics" : true,
    "positions": false,
    "offsets": false,
    "filter" : {
      "max_num_terms" : 3,
      "min_term_freq" : 1,
      "min_doc_freq" : 1
    }
}
```

Term Vectors

Example response:

```
{
   "_index": "imdb",
   "_type": "_doc",
   "_version": 0,
   "found": true,
   "term_vectors": {
      "plot": {
         "field_statistics": {
            "sum_doc_freq": 3384269,
            "doc_count": 176214,
            "sum_ttf": 3753460
         },
         "terms": {
            "armored": {
               "doc_freq": 27,
               "ttf": 27,
               "term_freq": 1,
               "score": 9.74725
            },
            "industrialist": {
               "doc_freq": 88,
               "ttf": 88,
               "term_freq": 1,
               "score": 8.590818
            },
            "stark": {
               "doc_freq": 44,
               "ttf": 47,
               "term_freq": 1,
               "score": 9.272792
            }
         }
      }
   }
}
```

References

[1] elastic.co. Elasticsearch reference [7.5]: Document apis. `https://www.elastic.co/guide/en/elasticsearch/reference/current/docs.html`, . Accessed: 2020-1-16.

[2] elastic.co. Elasticsearch reference [7.5]: Term vectors. `https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-termvectors.html`, . Accessed: 2020-1-16.