ElasticSearch Tutorial

Install ElasticSearch using following commands:

Exploring your cluster:

```
brew install elasticsearch@6.6.1
```

Install python libraries for elasticsearch using command:

```
pip install elasticsearch==6.3.1
```

Now start ElasticSearch server using the following command below:

```
elasticsearch
```

Now as the elasticsearch server is started now we can perform all the basic document operations like insertion, deletion, updation and querying using the elasticsearch api available. So first let us just walk through the indexing in the elasticsearch. First, the point is that in elasticsearch the data is stored in the form of indexes. So when hitting the api, we have to specify the index with which we are storing. If no index is provided then elasticsearch creates an automatic number which is very big in its own and hard to remember. So it is recommended to provide an index whenever you are inserting in elasticsearch.

Indexing API (It’s a put request):

PUT :- `index/_type/id`

Example:

```
got/_doc/1
{
      "user" : "Adam",
      "post_date" : "2019-04-15T14:12:12",
      "message" : "This season is really amazing"
}
```

Here the index is “got” and id of the inserted doc is 1. The user Adam has sent a message and we have stored it in the elasticsearch in that particular index. So we are storing in the form of a JSON file.

Similarly more data can be inserted in the elasticsearch using the following API.

```
got/_doc/2
{
      "user" : "Eve",
      "post_date" : "2019-04-16T56:12:12",
      "message" : "This season was really fantastic"
}
```

Get API (It’s a Get request):

Get: `index/_type/id`

Example:

```
got/_doc/1
```

The result would be something like:

```
{
    "_index" : "got",
    "_type" : "_doc",
    "_id" : "1",
    "_version" : 1,
    "_seq_no" : 10,
    "_primary_term" : 1,
    "found": true,
    "_source" : {
        "user" : "Adam",
        "date" : "2019-04-15T14:12:12",
        "likes": 0,
        "message" : “This season is really amazing"
    }
}
```

There is also one API to check whether with particular ID and Index, there exist any record or not. That particular API is:

```
HEAD index/_type/id
```

Example:

```
got/_doc/1
```

Delete API:

The delete API allows to delete a JSON document from a specific index based on its id.

```
Delete /index/_type/id
```

Example:

```
Delete got/_doc/1
```

The result of it would be somewhat like:

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

Update API:

The update API allows to update a document based on a script provided. The operation gets the document (collocated with the shard) from the index, runs the script (with optional script language and parameters), and indexes back the result (also allows to delete, or ignore the operation). It uses versioning to make sure no updates have happened during the "get" and "reindex".

```
POST /index/_type/id
```

Match Query API:

```
GET /_search
{
    "query": {
        "match" : {
            "message" : ""
        }
    }
}
```