Elasticsearch

NoSQL

A4- S8

ESILV
jihane.mali (at) devinci.fr

Chapter

Contents

Chapter 1

Setting Movies dataset

For this practice work, we will use the elasticsearch indexing software.

1.1 cURL

cURL is a small executable which sends/receives HTTP requests in command lines. You can specify headers by adding parameters to your command (XPUT/XGET/XPOST, -H "Content-Type...", –databinary...).

You can download cURL (already installed under Linux and MacOSX) here (better to use the SSL embedded version):
https://curl.haxx.se/download.html. It is natively installed on Linux, MacOSX and eventually Windows. the parameters are different on PowerShell, use "*Invite de commande*".

1.2 Elasticsearch & Kibana

The installation steps are a little bit straightforward[1]:

- Local installation: https://www.elastic.co/downloads
- Docker : name "elasticsearch-kibana" (@nshou v 6.5.4)

```
– docker pull nshou/elasticsearch-kibana:latest
```

  - For MacOSX with **M1 chipset** (2021), *amd64* VMs are not compatible. You should use *ARM64* VMs:
    https://www.docker.elastic.co/r/elasticsearch/elasticsearch-oss:7.10.2-arm64
    https://www.docker.elastic.co/r/kibana/kibana:7.13.4-arm64
    To achieve this, you should use a docker-compose to create a small cluster with both ES and Kibana[2].

1.3 Import dataset with cURL

- Download the movies_elastic.json.zip data file[3], and unzip it
- Import it in a **command line** with:

```
curl -XPUT localhost:9200/_bulk -H "Content-Type: application/json" --data-binary @movies_elastic.json
```

If it does not work, try :

```
curl --insecure -XPUT -u username:password https://localhost:9200/_bulk -H"Content-Type:
application/json" --data-binary @movies_elastic.json
```

where username and password are provided in your ElasticSearch container logs on Docker Desktop.

1.4 Test your database

1.4.1 Test on your browser, command line (cURL) or Kibana dev tools:

[1] A guide is online: https://chewbii.com/elasticsearch/
[2] An example here: https://chewbii.com/elasticsearch/
[3] https://devinci-online.brightspace.com/d2l/le/content/15379/viewContent/18377/View

```
GET http://localhost:9200/movies/1

GET http://localhost:9200/movies/_search
```

Chapter 2

DSL queries

For each query, you must provide an URL with :

- A server to reach: `https://localhost:9200`
- The index: `movies` (and for older ES versions "index/type")
- A used service: `_search`
- This produces the URL: `https://localhost:9200/movies/_search`

You can write the query in three formats (when possible):

- With the HTTP GET method (web brower), with the q= parameter

```
https://localhost:9200/movies/_search?q=title:foo+bar
```

- With the HTTP POST method, with the JSON document param *"query.json"* (called DSL queries):

```
curl -XPOST "localhost:9200/movies/_search&pretty" -H "Content-Type: application/json"
-d @query.json -o output.json
```

Again, if it does not work, try :

```
curl --insecure -XPOST -u username:password "https://localhost:9200/movies/_search?pretty" -H"Content-Type:application/json"
-d @query.json -o output.json
```

where username and password are provided in your ElasticSearch container logs on Docker Desktop.
And then open this JSON document in your Web browser.

- You can also make those queries directly in Kibana in the "Dev Tools"[1]

```
GET /movies/_search
{"query":{
      "match":{"title":"foo bar"}}
}
```

2.1 Simple queries

2.1.1 Every movies which title matches 'Star Wars' (match query),

2.1.2 Try with exact match (match_phrase),

2.1.3 Star Wars movies and Directors equal to 'George Lucas' (boolean query),

2.1.4 Movies were 'Harrison Ford' played,

2.1.5 Movies were 'Harrison Ford' played with a plot containing 'Jones',

2.1.6 Movies were 'Harrison Ford' played with a plot containing 'Jones' but plots without containing 'Nazis'

2.1.7 Movies of 'James Cameron' which rank is better than 1000 (boolean + range query)

2.1.8 Movies of 'James Cameron' which rating must be higher than 5 and which genre must not be 'Action' nor 'Drama'

2.1.9 Movies of 'J.J. Abrams' which released date is mandatory between 2010 and 2015 (**filtered** query on `release_date`)

[1] Kibana dev tool: `http://localhost:5601/app/dev_tools#/console`

2.2 Complex queries: Aggregate queries

We wish now to do some aggregate queries on the index in order to extract some statistics.

2.2.1 Give for each year the number of movies,

2.2.2 For each category (genres), give the number of movies. To take into account the whole text, you need to use "keyword" after the required field.

2.2.3 Give the average rating of movies,

2.2.4 Give the average rating of George Lucas' movies,

2.2.5 Count the number of movies for the given ranges of rating: 0-1.9, 2-3.9, 4-5.9...),

2.2.6 Number of distinct directors in adventures movies,

2.3 Hard queries: double aggregates & Mapping

2.3.1 Give the average rating per genre,

2.3.2 Give min, max and average rating for each genre,

2.3.3 Give the average ranking of movies per year and sort them increasingly,

2.3.4 Give average movie's rank and average movie's rating for each director. Sort the result decreasingly on average rating,

2.3.5 Give the terms occurrences extracted from each movie's title. The text value requires a specific mapping on the dataset stored in elasticsearch, see: `https://www.elastic.co/guide/en/elasticsearch/reference/current/fielddata.html`.

2.3.6 Most significant terms in plots of George Lucas movies,

Chapter 3

Bonus (More hard queries)

3.1 Synonyms

We can insert synonyms inside the search engine in order to change the mapping between words. For this, we need to define a new **analyzer**[1]

3.1.1 List of synonyms

First, create a file which contains the synonyms (called here *"SYNONYMS.TXT"*). The structure is detailed here: `https://www.elastic.co/guide/en/elasticsearch/guide/current/synonym-formats.html`

Here is an example:

```
"united states       => usa",
"united states of america => usa"
```

Put them in the following folder: `elasticsearch/config/analysis/SYNONYMS.TXT`

3.1.2 Change the analyzer

- Close your "movies" index:
  `POST movies/_close`
- Add a tokenizer:

```
PUT movies/_settings
{
   "settings": {
     "index" : {
       "analysis" : {
         "analyzer" : {
           "MY_SYNONYMS" : {
             "tokenizer" : "whitespace",
             "filter" : ["graph_synonyms"]
           }
         },
         "filter" : {
           "graph_synonyms" : {
             "type" : "synonym_graph",
             "synonyms_path" : "analysis/SYNONYMS.TXT"
           }
         }
       }
     }
   }
}
```

- Open your "movies" index:
  `POST movies/_open`

3.1.3 Query with synonyms

Use the analyzer

[1] `https://www.elastic.co/guide/en/elasticsearch/guide/current/using-synonyms.html`

```
POST movies/_search
{ "_source": {"includes":["..."]},
  "query": {
    "match" : {
      "fields.title": {
        "query" : "United States",
        "analyzer": "MY_SYNONYMS"
      }
    }
  }
}
```

To train, search for the *WordNet* dataset which contains a huge number of synonyms:
`https://www.kaggle.com/duketemon/wordnet-synonyms`
Change the file in the proper format, and import into ES. Then query your dataset with corresponding synonyms and see the different output.

3.2 Extra bonus

Other things you can look at:

- Use stemming and language analyzers:
  `https://www.elastic.co/guide/en/elasticsearch/reference/6.6/analysis-lang-analyzer.html`
- N-Grams tokenizers:
  `https://www.elastic.co/guide/en/elasticsearch/reference/6.6/analysis-ngram-tokenizer.html`
- Scripting with Painless:
  `https://www.elastic.co/guide/en/elasticsearch/painless/6.6/painless-examples.html`
- Add X-Pack plugins to elasticsearch:
  `https://www.elastic.co/guide/en/elasticsearch/reference/current/xpack-api.html`
- Integrate your **dataset** with *Logstash*:
  `https://www.elastic.co/guide/en/logstash/current/getting-started-with-logstash.html`