Méthodes d'accès aux données 2021–2022
Professeur : Nastaran Fatemi
Assistants : Christopher Meier
Edward Ransome

Getting started with *Elasticsearch*

1 Introduction

1.1 Objectives

The goal of this lab is to intall *Elasticsearch*, which is a distributed RESTful search engine used widely in the industry based on Apache Lucene (a free and open-source information retrieval software library), and to upload and parse a dataset .

1.2 Organization

There is no submission for this part, however all the steps are required to complete the next lab which will consist of indexing and searching the uploaded dataset.

1.3 Setup

First, download the archive from *Moodle* and extract it. It contains:

- `data/`
  - `cacm.txt` CACM collection, see 3.1.
  - `cacm.ndjson` CACM collection formated as line separated json objects for easy ingestion.
  - `common_words.txt` List of words that can commonly be removed (stop words, used in the next lab).
- `docker-compose.yml` correctly configured[1] docker deployment

Second, deploy an *Elasticsearch* cluster. The easiest way to do it is with docker.

With docker and docker-compose installed on your computer, it should be as simple as opening a terminal and executing `> docker-compose up -d` from inside the extracted archive.

You can now access the Kibana web interface via `http://localhost:5601`. Kibana is a free interface to visualize, navigate and manage data in *Elasticsearch*. To login use `elastic` as the username and `MAC2022` as the password.

1.3.1 Note

If you have this error in the logs, follow the instruction at this link:
`bootstrap check failure [1] of [1]: max virtual memory areas vm.max_map_count [XX] is too low`

You may need to do `> docker-compose down --volumes` before retrying to start the docker-compose. Attention, this will delete any data you may have stored in Elasticsearch.

[1] We have configured *Elasticsearch* so that:
- Kibana can securily communicate with *Elasticsearch*.
- At least one node in the cluster must have role `ingest`, it is the default.
- The file `common_words.txt` is available inside the config directory.

2 Familiarizing with *Elasticsearch*

2.1 Sending queries to *Elasticsearch*

You send data and other requests to *Elasticsearch* using REST APIs. This lets you interact with *Elasticsearch* using any client that sends HTTP requests, such as `curl`. We recommend to use *Kibana*'s console in the dev tools to send requests to *Elasticsearch*. This way you can easily take advantage of the API request examples present in the documentation.

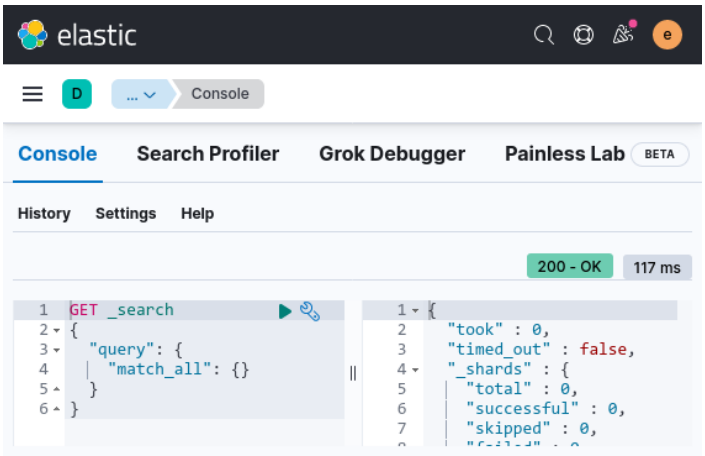

Figure 1: Screenshot of the dev console

When sending requests do not forget to escape special characters in JSON strings. An example of a string containing a new line is: `"Hello\nWorld"`.

Take a quick tour of the basic functionalities and queries by following the *Elasticsearch* Quick Start.

3 Ingesting and exploring the CACM collection

We are now going to use *Elasticsearch* to ingest and explore a list of scientific publications.

3.1 Description of the collection

In this document and the following lab we will use the famous text corpus, CACM. The CACM collection is a set of titles and abstracts from the journal Communications of ACM. It is provided in the file `cacm.txt`. To facilitate the ingestion, we also provide a transformed[2] version of CACM called `cacm.ndjson`. `cacm.txt` is provided only because it's easier to read.

Each line in `cacm.ndjson` is a JSON object with a `_row` attribute which contains the following information, separated by tabulations:

- the publication id
- the authors (if any, separated by ';')
- the title

[2] The content of `cacm.ndjson` has been generated using the following command:

```
> jq -c --raw-input --slurp 'split("\n") | map(select(. != "")) | {"_row": .[]}' cacm.txt
```

- the date of publication (year and month)
- the summary (if any)

There might be publications without any author or without the summary field.

3.2 Ingesting

The goal of this part is to upload and parse the CACM publication collection using Ingest pipelines.

You add data to Elasticsearch as JSON objects called documents. Elasticsearch stores these documents in searchable indices.

The file `cacm.ndjson` contains a version of the collection readable by *Kibana*.

1. Upload the collection into an index called `cacm_raw`. In Kibana, go to "Integrations > Upload file" and drop the `cacm.ndjson` file.
2. Create an ingest pipeline which parses the `_row` field and returns documents with only the following fields: `id`, `author`, `title`, `date`, `summary`.
   - You have two options: you can either manually create an HTTP request or you can use the Kibana interface (see detailed instructions in documentation).
   - In any case, you will need the following processors: csv[3] with quote set to "§"[4], split to separate authors and remove to delete the `_row` field.
3. In order to apply the created pipeline to the uploaded documents use reindex. Reindex copies the documents from a source index (in your case `cacm_raw`) into another index (here you call it `cacm_dynamic`) optionally using a pipeline.
4. Verify that you have the same number of documents to assure that reindex was correctly executed.

3.3 Exploring

The goal of this part is to explore the dataset and created index using Kibana and Data Views.

Kibana requires the creation of a Data View to discover the data inside the indexes.

1. Go to Stack Management > Data Views
2. Click on "Create data view".
3. Set the index pattern name to `cacm_dynamic`
4. Set the Time field to "I don't want to use the time filter"
5. Click on "Create data view"

Go to the Analytics > Discover panel and browse the collection using the new data view.

Find the answer to the following using KQL or "Field Statistics":

- What percentage of documents have a summary field?
- How many document have at least one author?
- How many document have been published after 1975?

You can also use it if you need to get the automatic `_id` assigned to a given document.

Note: Reindexing does not change the `_id` of a document.

[3] To insert a tab character in the browser, you need to type it in another program, for example a word processor, and then copy and paste it.
[4] The "§" character is not used in the dataset so we use it to avoid problems due to the presence of double quote in the dataset.