# Elasticsearch

## I) Introduction on Elasticsearch

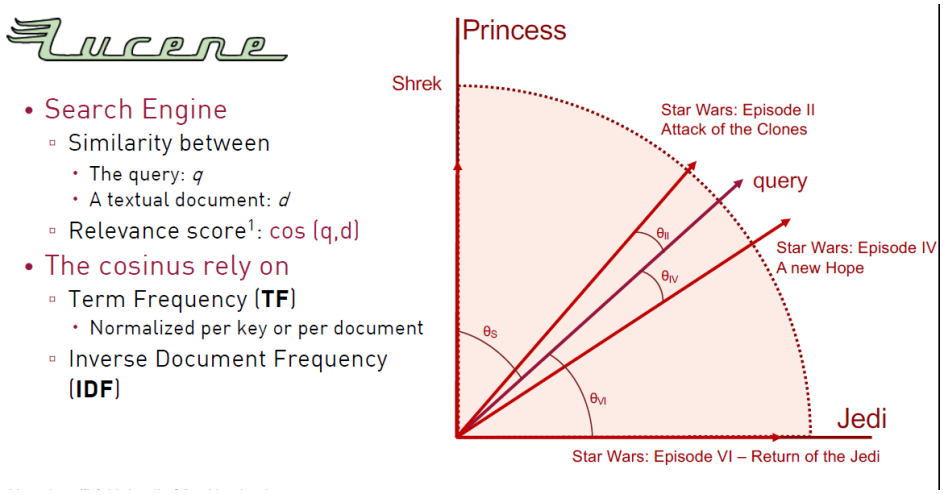
I chose to work on the Elasticsearch technology. Indeed, I like distributed data systems applied to decentralized databases (NoSQL[1]) which is one of my specialties.

**Principle:** Elasticsearch is a powerful search engine that as its name suggests is very manageable and flexible. It has been developed on Java and is based on the DSL[2] language. It is however multi-language and allows a deep search and analysis of even the most complex databases.

Unlike its counterparts MongoDB, which is simpler and more versatile, or Cassandra, which is more optimized for heavier data volumes, Elasticsearch has an outstanding search quality. For this, it relies on two principles:

- **indexing:** it makes it easier to find data. Indeed, with the help of these markers that serve as shortcuts and structuring within the data, the queries are executed efficiently

- **score:** this is based on the TF[3] that maximizes the score and the IDF[4] that minimizes the score. In this way, a score is assigned to the data in order to highlight those whose score is most relevant.




**Practical examples of uses:** Uber, Netflix, Shopify, Udemy, etc.

---

[1] **NoSQL**: databases that do not use traditional tables (rows and columns) for data storage. They organize large volumes of data using flexible techniques such as documents, charts, value pairs and columns.
[2] **DSL**: Domain Specific Langage which is a query langage.
[3] **TF**: term frequency.
[4] **IDF**: inverse document frequency.


**ELK[5] system:** Thus, Elasticsearch forms a trio with Logstash and Kibana.

Logstash is a data pipeline tool that is compatible with all services (S3, AWS, etc.).

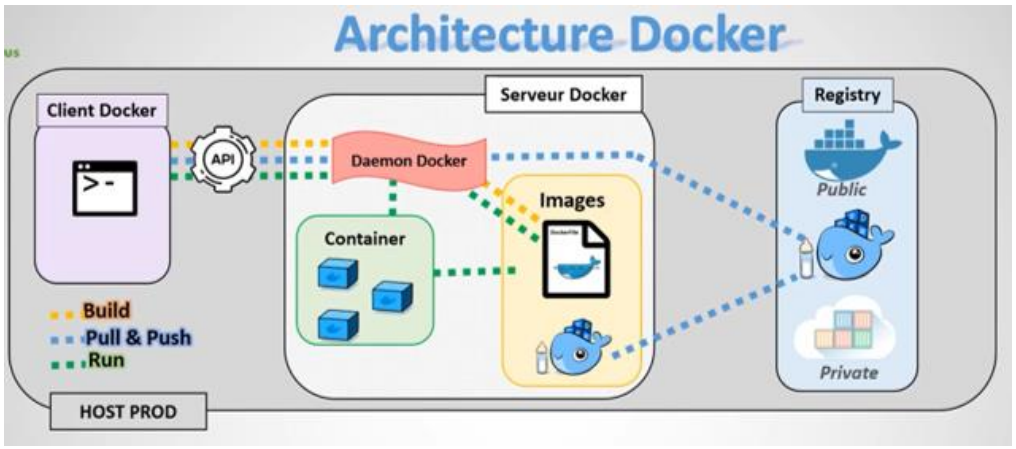
Kibana is a data visualization tool.

So, with this very complementary ELK system, we can import, manipulate and visualize even the most complex data efficiently.

# II) Docker

I chose to work through Docker. It's a controller based on a Linux[6] technology that allows you to create images and containers that work completely independently. Thus, one can create servers, connections (local) and use software in isolated environments and less impacted by the dependencies related to operating systems which allows to reduce data volumes, increase speed, portability, scalability, security and flexibility.

[5] **ELK:** Elasticsearch Logstash Kibana.
[6] **Linux:** OS exploitation system using Ubuntu technology.



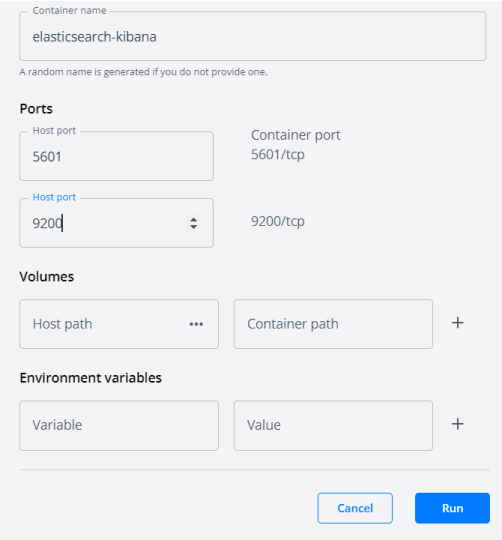

In this way, I avoid installing the necessary Elasticsearch which is quite large by passing through an image that I pull from Docker. I will use an image containing Kibana to simplify the writing of queries through its interface.

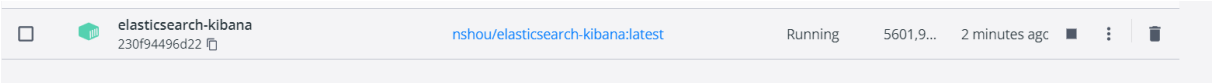
```
Microsoft Windows [version 10.0.19044.2006]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\hamza>docker pull nshou/elasticsearch-kibana:latest
latest: Pulling from nshou/elasticsearch-kibana
5ec7b7d162b2: Extracting [================>]  9.437MB/27.1MB
177617b11d13: Download complete
10273812b9e3: Download complete
ac553cdb1df6: Downloading [=====================================>]  33.03MB/42.4MB
1020ea78dce7: Download complete
2ea87ba2ee61: Downloading [=>]  10.74MB/486.8MB
```

After that, I configure 2 local ports (9200 for Elasticsearch and 5601 for Kibana).

| NAME ↑ | | TAG | IMAGE ID | CREATED | SIZE |
|---|---|---|---|---|---|
| alpine/git | IN USE | latest | 692618a0d74d | 20 days ago | 43.44 MB |
| docker101tutorial | IN USE | latest | f22d584f7541 | 18 minutes ago | 28.9 MB |
| nshou/elasticsearch-kibana | | latest | 27d008902117 | 9 months ago | 1.27 GB |






# III) Work on the dataset

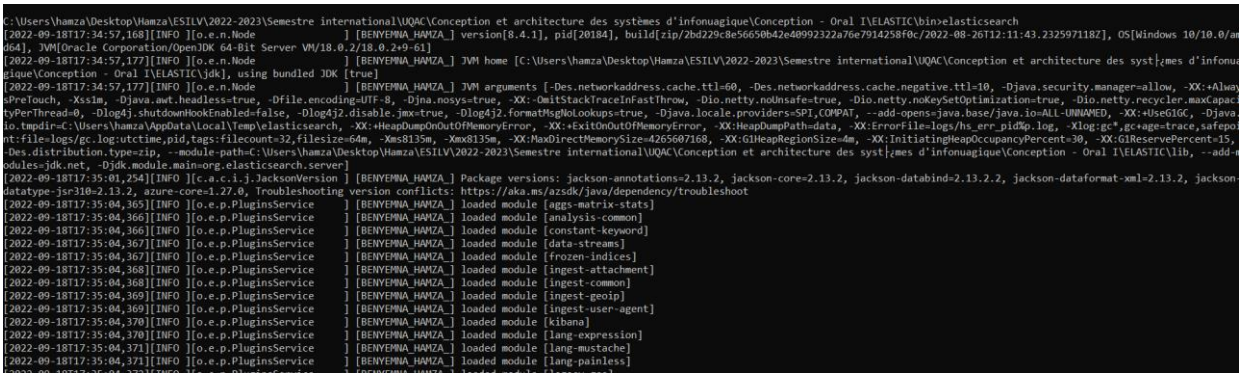
Once this is done, we can work on Elasticsearch through our localhosts. There are 3 different ways to import data:

- Logstash which I won't use.

- Curl[7] which allows you to use the RESTFul[8] API.

- Kibana.

## Curl: movies dataset



We launch the server then we import the database.

After that, we could make some query in the HTTP with a URL based syntax.

**Command: http://localhost:9200/movies/_search?q=title:Star+Wars**

[7] **Curl:** A HTTP request is made up of several components such as the URL to make the request to, HTTP verbs (GET, POST, etc.) and headers.

[8] **RESTFul API:** we could totally work on a localhost browser and make some query on the dataset (http://localhost:9200/).

```
▾ hits:
  ▾ total:
      value:                59
      relation:             "eq"
    max_score:              12.674389
  ▾ hits:
    ▾ 0:
        _index:             "movies"
        _type:              "_doc"
        _id:                "226"
        _score:             12.674389
      ▾ _source:
        ▾ directors:
            0:              "George Lucas"
          release_date:     "1977-05-25T00:00:00Z"
          rating:           8.7
        ▾ genres:
            0:              "Action"
            1:              "Adventure"
            2:              "Fantasy"
            3:              "Sci-Fi"
        ▾ image_url:        "http://ia.media-imdb.com/images/M/MV5BMTU4NTczODkwM15BMl5BanBnXkFtZTcwMzEyMTIyMw@@._V1_SX400_.jpg"
        ▾ plot:             "Luke Skywalker joins forces with a Jedi Knight, a cocky pilot, a wookiee and two droids to save the universe from the Empire's world-destroying battle-station, while also attempting to rescue Princess
                             Leia from the evil Darth Vader."
          title:            "Star Wars"
          rank:             226
          running_time_secs:  7260
        ▾ actors:
            0:              "Mark Hamill"
```


We could see the movie Star Wars with a max score of 12.67 because it is the exact name of the movie.

## Kibana: Basketwomen dataset

This time, I worked on Kibana which allows to write more or less complex queries with data import.

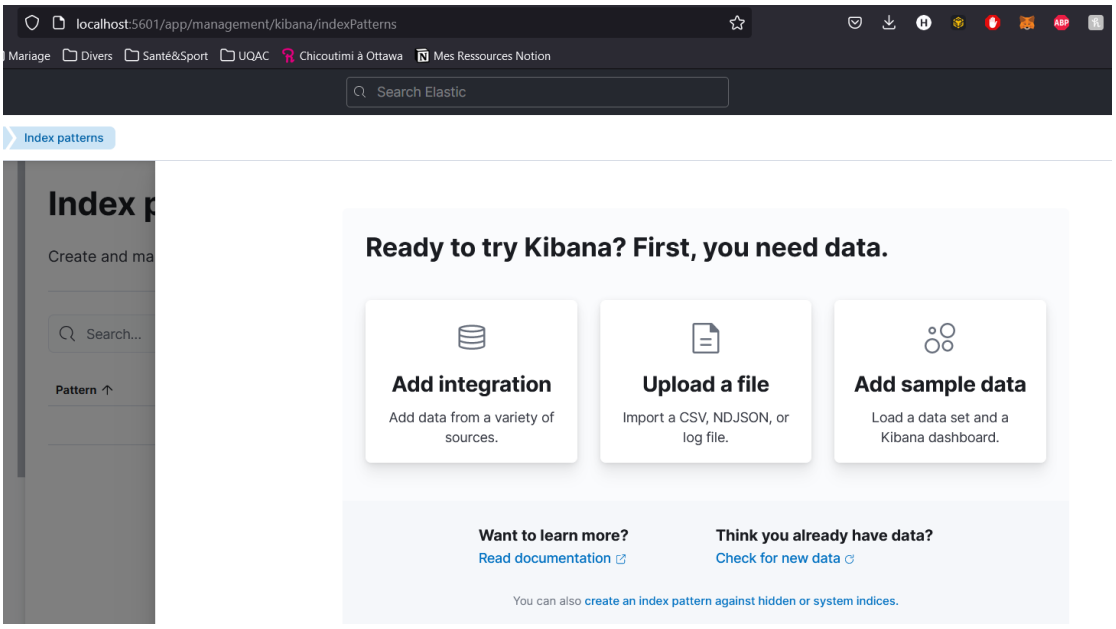
I first cleaned and corrected the format of the database required by Elasticsearch (*{"index": {"_index" : "INDEXNAME" "_id": X}})* using a Python program.

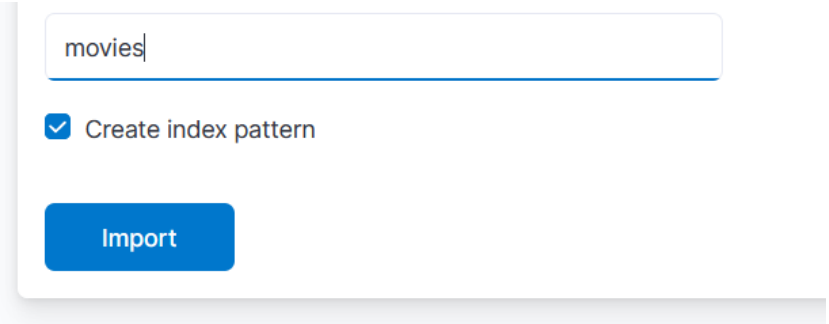
```python
# -*- coding: utf-8 -*-
"""
Created on Wed Jan 26 11:56:19 2022

@author: grego
"""


MyFile= open("Basketball_women.json",'r').read()
ClearData = MyFile.splitlines(True)
i=1
json_str=""
docs ={}
with open('outfile.json', 'w+') as outfile:
    for line in ClearData:
        #outfile.write(index_line+line)
        outfile.write('{"index": {"_index": "Players", "_type": "Player", "_id": '+str(i)+'}}\n'+line)
        i=i+1
```

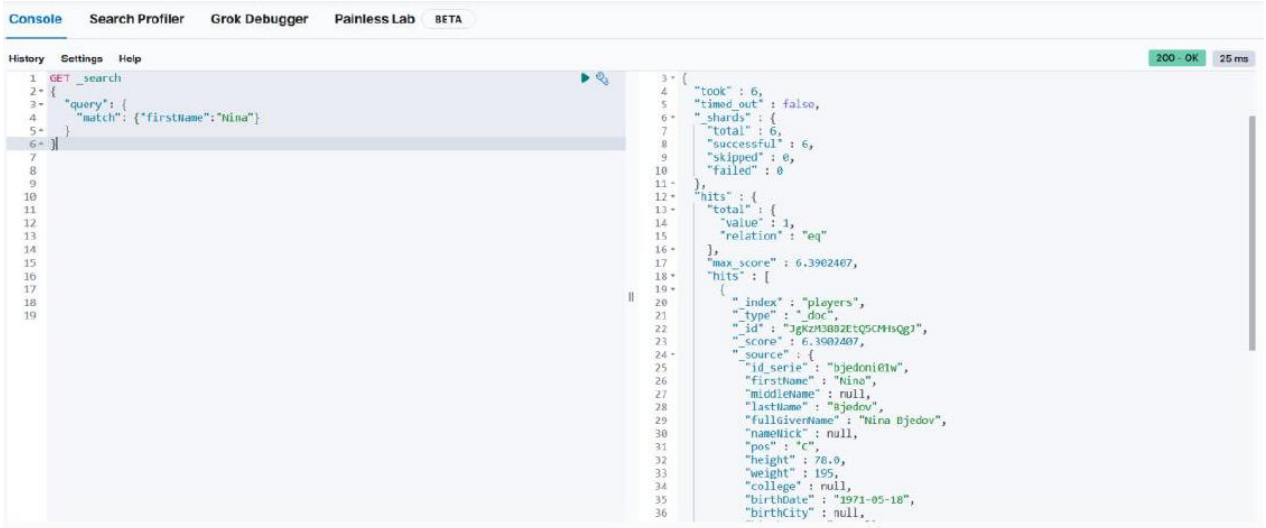
Then, we import it in Kibana by going to this URL:
[http://localhost:5601/app/management/kibana/indexPatterns](http://localhost:5601/app/management/kibana/indexPatterns)



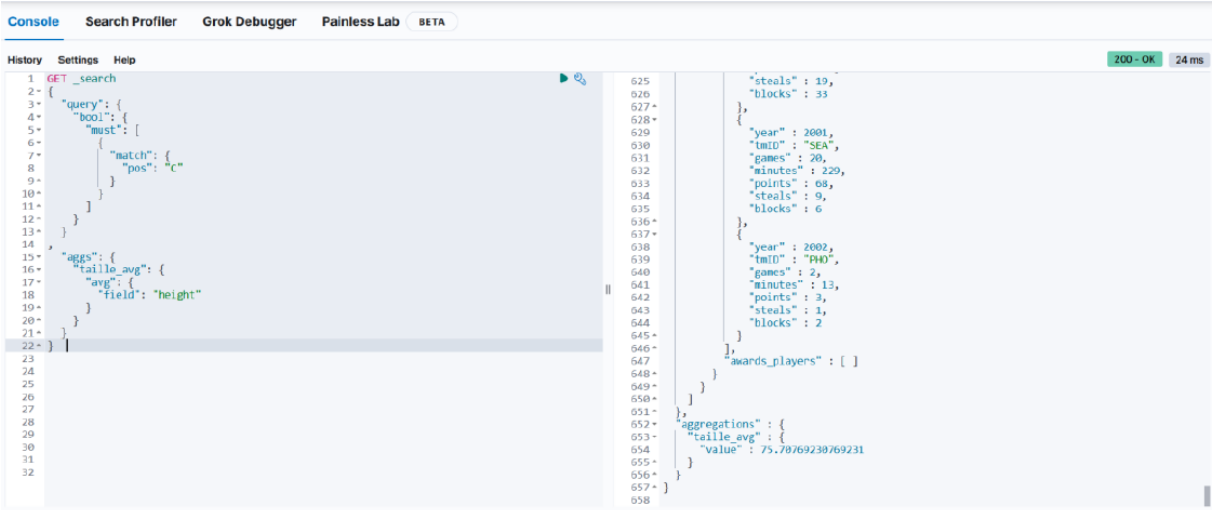




The import time was quite long and difficult, but the queries are more simple to do.

**<u>Query 1 (simple):</u> Basketball women named 'Nina'.**



**<u>Query 2 (difficult with aggregation[9]):</u> Average height of the basketball women playing as Center.**




# IV) <u>Conclusion:</u>

Assets:

Indeed, Elasticsearch is a reference in the world of search engines because it is RESTful API and therefore very robust and reliable, performing with its method of indexation and scoring very effective and allows many possibilities of database management (clusters, etc.) which makes it very scalable. We thus find a technology in which we find all the strengths of distributed systems in an optimal way.

Drawbacks:

However, it is not a good data host, especially in terms of security. The data I imported on Kibana was accessible to everyone and therefore not protected. Moreover, Elasticsearch requires data with a precise nomenclature without which the import of our data is simply impossible. Moreover, with the creation and verification of our database, the import is very long. Finally, it is a technology that takes up a lot of space. Even when importing an image via Docker where they are usually limited to a few MB, the Elasticsearch one I imported weighs more than 1Gb.

In conclusion, Elasticsearch is a great search engine tool that can be pushed more and has become a reference in the cloud and the NoSQL and cloud world.

[9] **Aggregation:** bring together distinct elements to form a homogeneous whole.

# Bibliography:

https://www.javatpoint.com/advantages-and-disadvantages-of-elasticsearch

https://www.access-it.fr/actu/elasticsearch-et-ses-avantages-pour-votre-erp/

https://www.compose.com/articles/how-scoring-works-in-elasticsearch/

https://www.elao.com/blog/dev/ameliorez-pertinence-resultat-elastic-search-score

https://www.youtube.com/watch?v=3hmA_K2MroY&ab_channel=DIGICACTUS

https://www.youtube.com/watch?v=8cH0ilGlQdE&ab_channel=InformatiqueSansComplexe

https://www.youtube.com/watch?v=_nmDN9nf1Lw&t=14s&ab_channel=xavki

https://chewbii.com/elasticsearch/

Elasticsearch.pdf, Nicolas Travel, copyright ESILV.