Labs
« ElasticSearch»

Prerquesite :

- Good Internet Connexion
- OS : Linux, MacOs, Windows 10
- YAML Editor : (VSCode, Atom, ...)
- Optional : Docker, Git

Lab0 : Installation

0.1 First start

Check your free disk space (you must have 20 % of your disk space free)

Download and unzip the latest release of Elastic Search

Start the server with $ES_HOME/bin/elasticsearch

If release is superior to 8, look at the trace and save all the information about password and enrollement token

Access to http(s)://localhost:9200

0.2 Configuration file and logs

Edit the main elasticsearch configuration file and modify the following properties:

- Cluster name
- Node name
- Listening address (put your public address there)

Try to start the server and observe the bootstrap checks traces, perform necessary fixes if necessary. (See https://www.elastic.co/guide/en/elasticsearch/reference/current/bootstrap-checks.html)

Change trace level to WARN

Set the ***node.name*** property via the command line when starting the server

0.3 Kibana Installation

Download a Kibana distribution with the same version number.

Unzip the archive and start Kibana (if 8.x +, with the enrollment token)

Access to http://localhost:5601 search for the *Dev Console*.

Execute the following queries :

```
GET /_cluster/health
GET /_search
GET /_cat/nodes
```

Lab1 (Optional) : Cluster, nodes, shards, replica

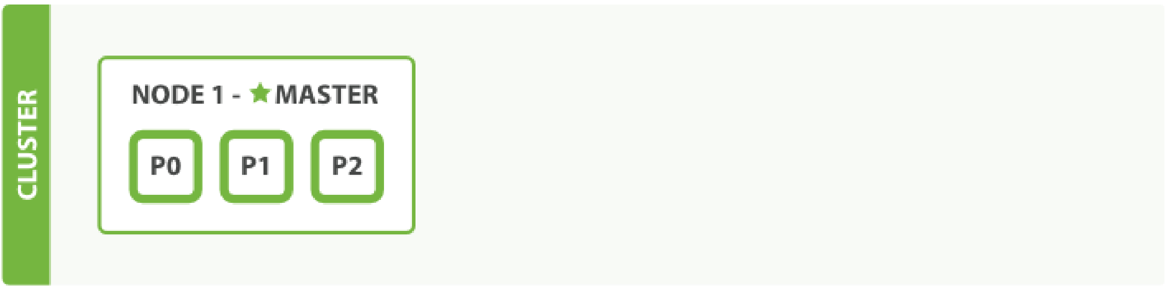
1.1 3 nodes cluster and shard replica

- Create an enrollment token with :
  bin/elasticsearch-create-enrollment-token -s node
- Uncomment the *transport.host* setting at the end of *config/elasticsearch.yml*.
- Restart Elasticsearch.

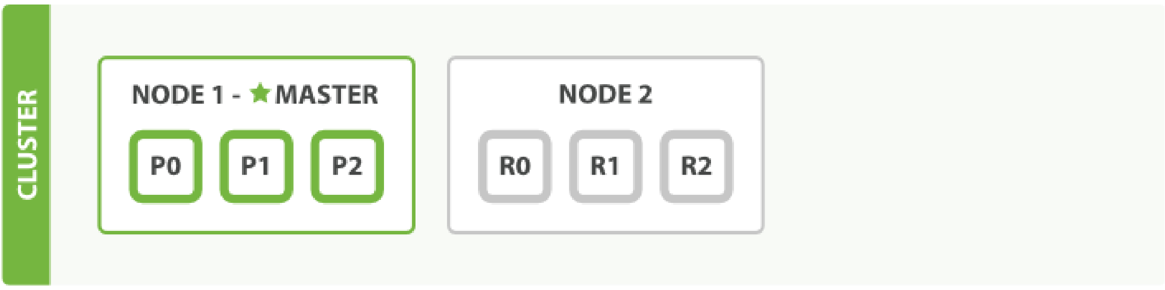
- Execute :
- `GET /_cluster/health?pretty`
  How many nodes, shards are available ?
- Create an index named *blogs*

```
PUT /blogs {
 "settings" : {
 "number_of_shards" : 3,
 "number_of_replicas" : 1
 }
}
```

- Re-execute *_cluster/health?pretty*
  Health status color ? How many shards available, active ?



- Unzip the distribution in another location
- Start a second node with the previous enrollment token :
  bin/elasticsearch --enrollment-token <token>
- Re-execute `_cluster/health?pretty`
  Health status color ? How many shards available, active ?



- Start a third node
  Health status color ? How many shards available, active ?
- Increase the number of replica

  ```
  PUT /blogs/_settings
   { "number_of_replicas" : 2 }
  ```
- Stop the first node
- Status health of the cluster ?
- Restrart the first node

1.2 Disabling security

For the remaining labs, we are disabling security. In 8.x+, security is enabled by default.

To disable it, you have to :

- Set `xpack.security.enabled: false`
- Comments all the properties relative to *xpack.security* in elasticsearch.yml
- Remove properties stored un elasticsearch keystore. You can do it with :

  ```
  bin/elastisearch-keystore remove <name-of-the-setting>
  ```

In *kibana.yml* comment all properties related to *xpack.security*

Lab2 : Document API

- Index the three provided JSON documents by the following curl commands or by Kibana
  `curl -XPOST /blogs/_doc/ --data-binary "@entry1.json"`
- Note the ids and retrieve document by ID
- Update 1 document by adding new fields and note the version increment:
  - *tags* : Array
  - *views* : Initialized to 0
- Perform updates with scripting:
  - increment a numeric field
  - Add an element to the tags array
  - Delete the *tags* field
- Delete index and perform previous indexing and updates commands via Bulk API

Lab3A : XML Ingestion with logstash

Objectives of this lab is to use a logstash pipeline and the ***xml filter*** in order to index XML content

3A.1 Installation of logstash

Download and unzip a distribution of ***logstash*** with the same version number as ElasticSearch and Kibana

Retreive the pipeline configuraion file ***xml.conf*** provided as a starting point.

Edit the file and change the ***path*** property according to your environment.

Execute ***bin/logstash -f <location_of_pipeline_conf>*** to test your installation.

After some times, you should see logs on the standard output.

3A.2 Use of XML filter

Look at the documentation of the XML filter and try to index the XML content provided into a single document in Elastic Search with all the fields you want

(See *https://www.elastic.co/guide/en/logstash/current/plugins-filters-xml.html)*

Lab3 : Ingestion of office documents

Installation of ingest-attachment plugin

- Install the ***ingest-attachment*** plugin
  ./elasticsearch-plugin install ingest-attachment
- Restart Node

Pipeline creation

- With the API, create a pipeline with a single processor *attachment*

Indexing

- Use the provided program to index all the provided documents :

- Check the number of indexed documents

Lab4 : Search Lite

Perform the following searches using the query string:

- Documents responding to "Java"
- Documents not responding to "Java"
- Limit the documents returned from the first request
- Documents whose content meets "Java"
- PDF documents with content responding to "Java"
- Documents with content that meets "Elastic Search"
- Documents whose title field contains administration
- Documents created after a particular date
- Documents created after a particular date and whose content matches "Java Elastic Search" but not "Administration"

Lab5 : Mapping and analyzers

5.1 Mapping and Analyzers

- Visualize the mapping for the index containing office documents. What are the full-text fields and what parsers are used?
- Test the differences between standard parser and French parser on French texts with REST requests. What are the stop-words used? How do words with accents, with apostrophes, compound words behave?
- Define the most appropriate mapping for office documents. The content field to be analyzed in French and in English.
- Perform identical searches on the 2 indexes and view the differences

5.2 Custom analyzer

- Create a new index for our document base with the following particularity:
- It uses specific parsers for English and French content:
  - increases the list of stop words
  - adds synonyms
- Redo indexing in this new index and perform searches

5.3 Reindexing API

Create a new index with a new settings for the shards and use the reindexing API to feed the new index.

Lab6 : Searching

6.1 DSL syntax

Perform DSL queries based on office index :

- PDF documents sorted by date
- Documents whose content field responds to "administration"
- Documents whose content or title field responds to "administration"
- Documents whose content or title field responds to "administration" and whose creation date falls within a range
- PDF documents whose content or title field responds to "administration" and whose creation date falls within a range
- Documents whose content field responds to "Administration" or "Oracle"
- Documents whose content field responds to "Administration" and optionally "Oracle"

6.2 Control relevance

Performing advanced searches with the ***explain*** parameter:

- Using ***boosting***, retrieve Documents whose *content* or *title* field responds to "administration". Documents with "administration" in the *title* field appearing first
- Same query using *should* clauses, compare scores
- Use another mode of calculation with ***dis_max***

6.3 Partial matching

- Prepare a new index that uses multiple indexing for the field *attachment.title* :
    - *keyword* DataType
    - *text* with *standard* analyzer
    - *text* with the *edge_ngram* analyzer
- Perform partial matching requests using one of the 3 mapping fields. Compare results and response times

6.4 Phrases

Perform queries with phrases:

- Retrieve documents containing the phrase "java framework", allow 5 word distance
- Retrieve documents whose title begins with "administration j"

6.5 Fuzzy, Natural language

- Perform fuzzy searches with typos
- <u>Optional :</u> Prepare a new index with a phonetic filter, perform searches with misspellings

6.5 Highlighting

Perform previous queries by adding highlight on content field (limit fragment size)

Lab7 : Agregations

- Execute agregation query :
  - By type of documents
  - By language
  - By both

- Define buckets by size of documents

- Find the average size of a document
- Find the averages size of document by type sorted with the highest value
- Average size by year
- Average size of PDF and .odp docs

Lab8 ; GeoQueries

Create a new index ***office-geo*** with one field set to the geo_point datatype :

```
PUT /office-geo
{
  "mappings": {
    "properties": {
      "location": {
        "type": "geo_point"
      }
    }
  }
}
```

Retreive the new java project wich ingest office document but with random location associated to each document

Execute it and perform different geo-queries :

- Filter around paris
- agregation around Paris