Cahier de TP
« ElasticSearch : Le moteur de recherche»

Pré-requis :
Poste développeur avec accès réseau Internet libre : Minimum 16Go RAM, 50Go Espace Disque
Linux (Recommandé) ou Windows 10
Pré-installation de :

- **Git**
- **JDK17**
- **Docker**
- **VisualStudio Code**

Table des matières

Atelier 1 : Mise en place

Télécharger les distributions de ElasticSearch et Kibana

1.1 Premier démarrage ElasticSearch

Dézipper l'archive dans un répertoire de travail référencé par la suite via $ES_HOME

Démarrer ElasticSearch

`$ES_HOME/bin/elasticsearch`

La commande affiche le mot de passe de l'utilisateur ***`elastic`*** et le jeton d'inscription pour Kibana.

Dans un autre terminal :

`export ELASTIC_PASSWORD="your_password"`

`curl --cacert config/certs/http_ca.crt -u elastic:$ELASTIC_PASSWORD https://localhost:9200`

1.2 Premier démarrage Kibana

Dézipper l'archive dans un répertoire de travail référencé par la suite via ***$KIB_HOME***

Démarrer Kibana

`$KIB_HOME/bin/kibana`

La console affiche un lien, l'ouvrir avec un navigateur et coller le jeton d'inscription

Connecter vous en utilisant le mot de passe de l'utilisateur ***elastic***

Une fois connecté vous pouvez changer le mot de passe de l'utilisateur elastic :
Management → Stack Management → Users

1.3 La Dev Console

Afficher la DevConsole

Management → Dev Tools

Effectuer les requêtes suivantes :

Santé du cluster :

```
GET /_cluster/health
```

API cat

```
GET /_cat/nodes
```

```
GET /_cat/indices
```

Recherche de tous les documents

```
GET /_search
```

Création d’index :

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

Ateliers 2: Document et indexation

2.1 Document API

Indexer les trois documents JSON fournis via la console Kibana

```
POST /blogs/_doc/
{
// Copié/collé des documents fournis
}
```

Noter les ids

Récupérer un document vis son ID

Effectuer une mise à jour partielle d'1 document en ajoutant 2 nouveaux champs :

- ***tags*** : Tableau initialisé à ['test']
- ***views*** : Initialisé à 0

Noter l'incrémentation de la version du document

Effectuer des mises à jour par script :

- incrémentation d'un champ numérique
- Ajout d'un élément dans le tableau tags
- Supprimer le champ tags

via l'API Bulk, exécuter en une requête :

- La création de 2 documents en précisant l'id
- la mise à jour partielle du 2 ème document via ***doc*** puis ***script***

2.2 Ingestion de documents via librairie cliente

2.2.1 Définition d'une pipeline

Définir une pipeline ayant comme 2 processeur :

- *attachment* sur un champ ***data***
- Suppression du champ ***data***

2.2.2 Indexation via librairie cliente

Se positionner dans le répertoire ***Tps/2.2_Ingestion/projects/ingest-<version>***

Vérifier les propriétés de connexion à Elastic Search dans le fichier
src/main/resources/application.properties
Construire l'application :

```
mvn clean package
```

Exécuter le programme
```
java -jar target/ingest-0.0.8-SNAPSHOT-jar-with-dependencies.jar
```

Indiquer les paramètres adéquats pour indexer le répertoire ../../doc

2.3 Search lite

Effectuer les recherches suivantes en utilisant la query string :

- Documents répondant à « Java »
- Documents ne répndant pas à « Java »
- Limiter les documents retournés de la première requête
- Documents dont le contenu répond à « Java »
- Documents PDF dont le contenu répond à « Java »
- Documents dont le contenu répond à « Elastic Search »
- Documents dont le champ titre contient administration
- Document créés après une date particulière
- Document créés après une date particulière et dont le contenu répondant « Java Elastic Search » mais pas « Administration »

Ateliers 3 : Mapping et analyseurs

3.1 Test des analyseurs

Tester les différences entre analyseur standard et analyseur français sur des textes en français avec des requêtes REST.

Comment se comporte les mots avec accents, avec apostrophes, les mots composés ?

3.2 Mapping explicite

Visualisez le mapping pour l'index contenant les documents bureautiques. Quel sont les champ full-text et quels analyseurs sont utilisés ?

Définir le mapping le plus approprié pour les documents bureautique. Le champ *content* devant être analysé en français et en anglais.

Faire des recherche identiques sur les 2 index et visualiser les différences

3.3 : Analyseur Custom

Tokenizer reference

https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-tokenizers.html

Token filter reference

https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-tokenfilters.html

Texte à indexer

```
"Un <b>hamburger</b>, parfois hambourgeois (au Canada francophone)
ou par aphérèse burger, est un sandwich d'origine allemande, composé de
deux
pains de forme ronde (bun) parfois garnis de viande hachée (souvent du
bœuf)
et généralement de crudités — <em>salade</em>, <em>tomate</em>,
<em>oignon</em>, <em>cornichon</em> (pickles) —,
de fromage et de sauce. C'est un plat typique de la restauration rapide,
emblématique de la cuisine américaine."
```

3.3.1 Filtre de caractères

Supprimer les balises HTML et convertir les entités HTML

```
POST /_analyze
{
```

```
  "char_filter": ["html_strip"],
  "text": "Un <b>hamburger</b>, parfois hambourgeois (au Canada
francophone) ou par aphérèse burger, est un sandwich d'origine allemande,
composé de deux pains de forme ronde (bun) parfois garnis de viande hachée
(souvent du bœuf) et généralement de crudités — <em>salade</em>,
<em>tomate</em>, <em>oignon</em>, <em>cornichon</em> (pickles) —, de
fromage et de sauce. C'est un plat typique de la restauration rapide,
emblématique de la cuisine américaine."
}
```

3.3.2 Tokenizer

Ajout du tokenizer `standard`

```
POST /_analyze
{
  "char_filter": ["html_strip"],
  "tokenizer": "standard",
  "filter": [
    "lowercase"
  ],
  "text": "Un <b>hamburger</b>, parfois hambourgeois (au Canada
francophone) ou par aphérèse burger, est un sandwich d'origine allemande,
composé de deux pains de forme ronde (bun) parfois garnis de viande hachée
(souvent du bœuf) et généralement de crudités — <em>salade</em>,
<em>tomate</em>, <em>oignon</em>, <em>cornichon</em> (pickles) —, de
fromage et de sauce. C'est un plat typique de la restauration rapide,
emblématique de la cuisine américaine."
}
```

3.3.2 Analyseur custom

Créer un analyseur custom nommé ***my_custom_analyzer***

```
PUT /analyzer_test
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_custom_analyzer": {
          "type": "custom",
          "char_filter": ["html_strip"],
          "tokenizer": "standard",
          "filter": [
            "lowercase"
          ]
        }
      }
    }
  }
}
```

Test de l’analyseur custom

```
POST /analyzer_test/_analyze
{
  "analyzer": "my_custom_analyzer",
  "text": "Un <b>hamburger</b>, parfois hambourgeois (au Canada
francophone) ou par aphérèse burger, est un sandwich d'origine allemande,
composé de deux pains de forme ronde (bun) parfois garnis de viande hachée
(souvent du bœuf) et généralement de crudités — <em>salade</em>,
<em>tomate</em>, <em>oignon</em>, <em>cornichon</em> (pickles) —, de
fromage et de sauce. C'est un plat typique de la restauration rapide,
emblématique de la cuisine américaine."
}
Step 6 :
```

Supprimer

```
DELETE /analyzer_test
```

3.3.4 Stop words

Enlever les stopwords français

french_stop, qui retire les tokens tels que **en**, **au**, **du**, **par**, **est**…

https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-stop-tokenfilter.html#analysis-stop-tokenfilter-stop-words-by-lang

```
PUT /analyzer_test
{
  "settings": {
    "analysis": {
      "filter": {
        "french_stop": {
          "type": "stop",
          "stopwords": "_french_"
        }
      },
      "analyzer": {
        "my_custom_analyzer": {
          "type": "custom",
          "char_filter": ["html_strip"],
          "tokenizer": "standard",
          "filter": [
            "lowercase",
            "french_stop"
          ]
        }
      }
    }
  }
}
```

Test puis suppression

3.3.5 Stemming

french_stemmer applique une racinisation (stemming) de nos tokens, c'est ce qui permet de supprimer les formes plurielles, les différentes conjugaisons, accord de genre sur un mot.

https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-stemmer-tokenfilter.html#analysis-stemmer-tokenfilter-language-parm

```
PUT /analyzer_test
{
  "settings": {
    "analysis": {
      "filter": {
        "french_stop": {
          "type": "stop",
          "stopwords": "_french_"
        },
        "french_stemmer": {
          "type": "stemmer",
          "language": "light_french"
        }
      },
      "analyzer": {
        "my_custom_analyzer": {
          "type": "custom",
          "char_filter": ["html_strip"],
          "tokenizer": "standard",
          "filter": [
            "lowercase",
            "french_stop",
            "french_stemmer"
          ]
        }
      }
    }
  }
}
```

Test puis suppression

3.3.6 Filtre elipson

french_elision, il enlève les articles pouvant précéder un mot, et donc **d'origine** devient **origine**.

```
PUT /analyzer_test
{
  "settings": {
    "analysis": {
      "filter": {
        "french_elision": {
```

```
          "type":         "elision",
          "articles_case": true,
          "articles": [
              "l", "m", "t", "qu", "n", "s",
              "j", "d", "c", "jusqu", "quoiqu",
              "lorsqu", "puisqu"
            ]
        },
          "french_stop": {
          "type": "stop",
          "stopwords": "_french_"
        },
        "french_stemmer": {
          "type": "stemmer",
          "language": "light_french"
        }
      },
      "analyzer": {
        "my_custom_analyzer": {
          "type": "custom",
          "char_filter": ["html_strip"],
          "tokenizer": "standard",
          "filter": [
            "french_elision",
            "lowercase",
            "french_stop",
            "french_stemmer"
          ]
        }
      }
    }
  }
}
```

3.3.7 Filtre synonymes

L'ajout de synonymes est aussi à considérer : il serait tout à fait intéressant que « salade » puisse être trouvé en recherchant « laitue », c'est le rôle du filtre **synonym**.

```
PUT /analyzer_test
{
  "settings": {
    "analysis": {
      "filter": {
        "french_elision": {
          "type":         "elision",
          "articles_case": true,
          "articles": [
              "l", "m", "t", "qu", "n", "s",
              "j", "d", "c", "jusqu", "quoiqu",
              "lorsqu", "puisqu"
            ]
        },
```

```
          "french_stop": {
          "type": "stop",
          "stopwords": "_french_"
        },
        "french_stemmer": {
          "type": "stemmer",
          "language": "light_french"
        },
        "french_synonym": {
          "type": "synonym",
          "expand": true,
          "synonyms": [
            "salade, laitue",
            "mayo, mayonnaise",
            "grille, toaste"
          ]
        }
      },
      "analyzer": {
        "my_custom_analyzer": {
          "type": "custom",
          "char_filter": ["html_strip"],
          "tokenizer": "standard",
          "filter": [
            "french_elision",
            "lowercase",
            "french_stop",
            "french_stemmer",
            "french_synonym"
          ]
        }
      }
    }
  }
```

3.4 API de réindexation

Créer un nouvel index avec un nouveau settings pour les shards, un alias et un nouveau mapping avec comme particularité :

- Il n'accepte pas le *dynamic mapping*
- Il utilise des analyseurs spécifiques pour le contenu anglais et français:
  - augmente la liste des stop words
  - ajoute des synonymes

Utiliser l’API de réindexation pour alimenter le nouvel index.

Ateliers 4 : Requêtes DSL

4.1 Combinaisons de clauses

Effectuer les requêtes DSL sur la base documentaire :

- Document PDF triés par date
- Documents dont le champ *content* répond à « administration »
- Documents dont le champ *content* ou *title* répond à « administration »
- Documents dont le champ *content* ou *title* répond à « administration » et dont la date de création est comprise dans un intervalle
- Documents PDF dont le champ *content* ou *title* répond à « administration » et dont la date de création est comprise dans un intervalle
- Documents dont le champ *content* répond à « Administration » ou à « Oracle »
- Documents dont le champ *content* répond à « Administration » et éventuellement à « Oracle »

4.2 Contrôle du score

Effectuer des recherches avancée avec le paramètre explain :

- En utilisant le boosting, récupérer les Documents dont le champ content ou title répond à « administration ». Les documents ayant « administration » dans le champ title apparaissant en premier
- Même requête en utilisant des clauses should, comparer les scores
- Utiliser un autre mode de calcul avec *dis_max*

4.3 Matching partiel

- Préparer un nouvel index qui utilise une indexation multiple pour le champ *attachment.title* :
  - Type *keyword*
  - Type *text* avec standard analyseur
  - Type *text* avec un analyseur utilisant *edge_ngram*
- Effectuer des requêtes de matching partiel en utilisant un des 3 champs du mapping. Comparer les résultats et les temps de réponse

4.4 Phrases

Effectuer des requêtes avec des phrases :

- Récupérer les documents contenant la phrase « framework java », autoriser une distance de 5 mots
- Récupérer les documents dont l titre commence par « administration j »

4.5 Fuzzy, Langage naturel

- Effectuer des recherches fuzzy avec des fautes de typo

- Optionnel : Préparer un nouvel index avec un filtre phonétique, effectuer des recherches avec des fautes d’orthographe

4.6 Surbrillance

Effectuer les requêtes précédentes en ajoutant la surbrillance sur le champ content (limiter la taille du fragment)

Atelier 5 : Agrégations

- Effectuer des agrégations de comptage
  - par type de documents
  - par langue
  - par les 2

- Regroupement par taille

- Trouver la taille moyenne d'un document
- Trouver la taille moyenne d'un document par type ordonné par la plus haute valeur
- Taille moyenne par année
- Taille moyenne des documents PDF et des documents odp

Atelier 6 : Géolocalisation

Dans ce TP, nous mettons en place une pipeline destinée à traiter les traces d'accès d'un serveur web. La pipeline utilisera entre autres le processeur *geo_ip* qui géolocalise à partir d'une adresse IP.

Une fois la pipeline mise en place, nous ingérons un fichier de log complet et effectuerons des requêtes d'agrégations

Mise en place de la pipeline :

```
PUT _ingest/pipeline/access_log
{
  "description" : "Ingest pipeline for Combined Log Format",
  "processors" : [
    {
      "grok": {
        "field": "message",
        "patterns": ["%{IPORHOST:clientip} %{USER:ident} %{USER:auth} \\[%
{HTTPDATE:timestamp}\\] \"%{WORD:verb} %{DATA:request} HTTP/%
{NUMBER:httpversion}\" %{NUMBER:response:int} (?:-|%{NUMBER:bytes:int}) %
{QS:referrer} %{QS:agent}"]
      }
    },
    {
      "date": {
        "field": "timestamp",
        "formats": [ "dd/MMM/YYYY:HH:mm:ss Z" ]
      }
    },
    {
      "geoip": {
        "field": "clientip"
      }
    },
    {
      "user_agent": {
        "field": "agent"
      }
    }
  ]
}
```

Tester la pipeline avec :

```
POST _ingest/pipeline/access_log/_simulate
{
  "docs": [
    {
      "_source": {
        "message": "212.87.37.154 - - [12/Sep/2016:16:21:15 +0000] \"GET
/favicon.ico HTTP/1.1\" 200 3638 \"-\" \"Mozilla/5.0 (Macintosh; Intel Mac
OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116
Safari/537.36\""
```

```
        }
      }
    ]
}
```

Utiliser le programme fourni qui lit chaque ligne du fichier de log fourni et le fournit à elastic search dans un index nommé ***apache_access***

Visualiser l'index et écrire :

- des requêtes de type filtre par *geo_bounding_box*
- des requêtes d'agrégations par *geo_distance*