Nom du professeur : Patrice Bellot **Recherche d'information et Recommandation TP 1**

Expérimentation avec ElasticSearch

Objectifs et questions :

- Comprendre les formats de documents existants et les méta-données associées
  - constituer un corpus de documents PDF, XML et JSON à partir du site ISTEX.fr
    - sur la page d'accueil vous trouverez un lien pour cela : choix "Chercheur" puis "Télécharger un corpus"
    - saisir une requête avec un ou deux mots clés et choisir Texte intégral pour avoir tous les formats dont JSON et TEI / XML (ce dernier se trouve dans le .zip).
    - NB : Votre corpus doit contenir au moins une dizaine de documents. Mais n'en prenez pas 100 000 non plus. Vous pouvez limiter le nombre de documents depuis l'interface du site Web ISTEX.
    - Chaque document est dans un répertoire séparé.
    - Par ailleurs vous devez vous connecter en utilisant vos identifiants AMU pour accéder aux fichiers.
  - Étudier les méta-données disponibles dans les versions XML et JSON
- Indexer et requêter des documents avec la Console d'Elastic Search
  - Indexer les documents JSON avec Elasticsearch.
  - Créer et tester des requêtes
  - Indexer des PDF :
    - Tester des utilitaires du type pdf2text et analyser les sorties obtenues : quels sont les éléments et les contenus qui sont mal identifiés ?
    - indexer des PDF (voir aide plus bas)
    - tester des requêtes sur les PDF
- Explorer les autres fonctionnalités de Kibana notamment pour l'analyse de fichiers logs.
  - Accessible depuis la fenêtre de démarrage :"Try our sample data"
    - choisir un jeu de données
    - essayer les différents modules accessibles depuis la fenêtre de démarrage dans le menu Kibana (en commençant par Discover, pour voir les données puis Dashboard et Maps)

Aide :

si vous souhaitez télécharger Elastic et Kibana : https://www.elastic.co/fr/start

Expérimentation sur 36 milliards de documents : https://thoughts.t37.net/how-we-reindexed-36-billions-documents-in-5-days-within-the-same-elasticsearch-cluster-cd9c054d1db8

pour démarrer les 2 serveurs :

1. aller dans le répertoire où la suite Elastic Search a été téléchargée ou déplacée (par exemple dans ~/Téléchargements)
2. ouvrir un Terminal puis :
    - dans le répertoire "elasticsearch-7.10.0" faire `bin/elasticsearch`
    - dans le répertoire kibana-7.10.0..... faire
    - bin/kibana
    - !!! Si vous avez un message d'erreur lié à Java c'est que la localisation de Java n'est pas standard. Il faut alors positionner la variable d'environnement JAVA_HOME à la bonne adresse
        - sous Linux ou Mac OS : en tapant la commande suivante (en adaptant bien sûr le nom du dossier)
          `export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64`
        - sous Windows en allant dans le Panneau de configuration, Paramètres Systèmes avancés, Variables d'environnement / Variables système
        - vous pouvez vérifier où se trouve Java sur votre machine en tapant la commande suivante dans un terminal ou Cygwin: `which java`
3. ouvrir l'interface graphique Kibana avec un navigateur Web en allant à :
    1. http://localhost:5601 (le port 5601 est le port par défaut de l'interface graphique Kibana ; le port par défaut du moteur Elastic Search est 9200, il sera utilisé par la suite avec les commandes curl, attention de ne pas confondre)
    2. Choisir dev_tools

aide générale en ligne : https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html

De façon générale il y a 3 manières d'utiliser Elastic Search une fois que les serveurs sont lancés :

- depuis un navigateur Web via l'interface graphique Kibana en utilisant des commandes GET / POST
- depuis un Terminal en s'adressant au serveur Kibana via la commande curl et les commandes GET / POST
- depuis un programme (Python, Java, ...) qui communique avec l'OS via des commandes curl comme ci-dessus ou via une API dédiée

En outre, une liste de commandes GET / POST peut être réunie dans un fichier script qui sera exécuté en une seule fois via une commande _bulk (voir ci-dessous). Utile lorsque l'on souhaite indexer une grande quantité de fichiers à la suite.

INDEXATION :

https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started-index.html

création d'un index vide avec le nom "monindex" :

```
curl -H "Content-Type: application/json" -XPUT "localhost:9200/monindex?pretty"
```

NB : 9200 est le numéro de port par défaut pour le serveur Elastic Search

NB : cette commande est en fait optionnelle : lorsque vous rajouter un document en indiquant un nom d'index qui n'existe pas, alors un nouvel index est automatiquement créé.

vérification index bien créé :

- dans l'interface Web Kibana (barre du haut) :
    - Soit choisir Management/Stack Management puis Data / Index Management (permet de voir les paramètres, de supprimer l'index etc.)

- Soit choisir la console DevTools puis saisir la commande ci-dessous et l'exécuter en cliquant sur la petite flèche à droite de la commande saisie `GET /monindex`

  ou

  `GET /monindex/_search`

- `curl -H "Content-Type: application/json" -XGET "localhost:9200/monindex?pretty"`

indexation d'UN SEUL contenu JSON, directement dans la requête :

tel que celui donné par ISTEX (exemple : 0C859E527FC68912C084A1F2BD5487D9645CF29F.json) en copiant son contenu et avec génération automatique de l'id :

POST /monindex/_doc/

{ ... tout le contenu du JSON qui a été copié-collé mais sur UNE SEULE LIGNE .... }

pour indexer UN ou PLUSIEURS fichiers JSON :

il faut :

- soit utiliser autant de fois que de fichiers (une commande pour chaque fichier .json):

```
curl -H "Content-Type: application/json" -XPOST "localhost:9200/monIndex/_doc/" -d '{ ..... le contenu JSON sur 1 ligne..... }'
```

NB : attention il ne faut pas de saut de ligne dans la commande... (modifier à la main ou bien utiliser avant un pré-processeur de JSON comme jq -- avec la commande (voir https://dev.to/ddeveloperr/how-to-convert-json-to-ndjson-3373):

```
cat XXX.json | jq -c > XXXNDJSON.json
```

- soit utiliser une seule commande _bulk qui pointe vers un fichier commandes.json contenant lui-même les actions à effectuer sur chaque document .json. Il y a dans ce fichier autant de lignes que de fichiers .json :

```
curl -H "Content-Type: application/json" -XPOST "localhost:9200/monindex/_doc/_bulk?pretty&refresh" --data-binary "@commandes.json"
```

voir : https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html

et https://qbox.io/blog/understanding-bulk-indexing-in-elasticsearch

avec le fichier commandes.json de la forme (1 ligne pour la commande, 1 ligne pour le contenu et ainsi de suite)

```
{ "index": {"_index":"monindex","_type":"doc"}}
{ ..... contenu JSON fichier 1 .... }
{ "index": {"_index":"monindex","_type":"doc"}}
{ ..... contenu JSON fichier 2 .... }
```

NB:

- !!! bien penser à ajouter une ligne vide à la fin du fichier)
- vous devez être dans le dossier qui contient commands.json avant d'exécuter la commande (pour Windows avec Cygwin, utiliser la commande cd C:/..../..../..../ --> attention de bien respecter le sens des / et si votre chemin d'accès contient un espace alors faire précéder cet espace par un \)

pour parcourir des répertoires et indexer tous les fichiers qui s'y trouvent :

s'inspirer du script en Python : https://gist.github.com/stevehanson/7462063. Vous pouvez le reprendre en le modifiant :

- en début de code (ligne 6 INDEX=....), indiquer le nom de votre index à la place du nom présent
- si vous ne changez pas les noms de dossiers, alors mettre le code Python dans le même dossier que celui qui contient vos documents à indexer

- modifier la liste des types de fichiers que vous souhaiter indexer
    - par exemple rajouter le type 'json' dans la liste des types INDEX_FILE_TYPES
    - si vous indexez des fichiers JSON avec ce script, il faut alors supprimer l'appel à la méthode createEncodedTempFile(fname)
    - pour tous les autres formats, la méthode createEncodedTempFile(fname) doit être appelée
- pour que ce soit compatible avec ElasticSearch 6.x et suivants, dans la méthode postFileToIndex:
    - rajouter l'option suivante dans l'appel curl
    - `-H "Content-Type: application/json"`
    - remplacer -X PUT par. -XPOST
- pour lancer le script, utiliser la commande python avec le nom du script. Le chemin d'accès à vos documents vous sera alors demandé.
- pour les formats PDF, .docx, .ppt etc. ce script fait appel à un plug-in qu'il faut installer (pour Elastic 6.x, The mapper attachments plugin qui est décrit ici https://qbox.io/blog/powerful-pdf-search-elasticsearch-mapper-attachment ; pour Elastic Search 7.x remplacé par l'Ingest Attachment, voir section suivante)

pour indexer des documents PDF ou .docx :

- vous pouvez les convertir en JSON
    - pour les PDF, il faut d'abord en extraire le texte puis le formatter en JSON. Un utilitaire comme PDF2JSON fait cela (https://github.com/flexpaper/pdf2json) mais il en existe d'autres, certains en ligne (exemple PDFMaLL). Chacun est plus ou moins capable d'extraire correctement le texte, les colonnes et les tableaux. L'autre possibilité est de convertir d'abord en texte puis en JSON mais on perd alors toute structure.
    - les fichiers .docx sont en fait une archive compressée de fichiers XML... Pour y accéder il faut "dézipper" le fichier .docx et l'on obtient des fichiers pour le contenu, l'en-tête etc. qu'il ne reste qu'à regrouper et à convertir en JSON, par exemple avec l'outil de commande en ligne xq (voir un exemple d'usage de xq)
- on peut utiliser le plug-in Ingest Attachment (droits administrateur nécessaires) :
    - https://www.elastic.co/guide/en/elasticsearch/plugins/current/ingest-attachment.html, voir aussi https://kb.objectrocket.com/elasticsearch/how-to-index-a-pdf-file-as-an-elasticsearch-index-267
    - on peut utiliser FSCrawler : https://fscrawler.readthedocs.io/en/fscrawler-2.5/index.html qui utilise ce plug-in mais qui rajoute en plus la possibilité de définir un dossier "documents" dynamique (exploré à intervalles réguliers, les nouveaux documents sont alors automatiquement rajoutés dans l'index).
    - les fichiers à indexer doivent être convertis en base 64 (voir https://fr.wikipedia.org/wiki/Base64). Cela peut être fait par la commande unix base64 ou bien par la commande python encode("base64") ou base64.b64encode (https://docs.python.org/2/library/base64.html). Pour un exemple d'utilisation permettant de voir les contenus lors de la recherche (à la place du code base64) : https://qbox.io/blog/index-attachments-files-elasticsearch-mapper

REQUETAGE (recherche de documents):

- pour créer des requêtes :

pour démarrer : https://www.elastic.co/guide/en/elasticsearch/reference/6.0/_the_search_api.html

ensuite : https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-body.html

exemple de requêtes :

- `GET /monindex/_search?q=cognitive`
- `GET /monindex/_search?q=title:cognitive`
  `ou`
  `GET /monindex/_search`
- `{`
- `"query": { "match": {"title":"cognitive"} }`
- `}`

(voir https://www.elastic.co/guide/en/elasticsearch/reference/6.0/_executing_searches.html)

- pour créer des synthèses de l'index ou des documents trouvés ou filtrer ces derniers , utiliser des agrégations :

https://www.elastic.co/guide/en/elasticsearch/reference/6.2/_executing_aggregations.html

https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations.html

https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket-terms-aggregation.html

voir l'exemple : https://netapsys.developpez.com/tutoriels/nosql/agregations-elasticsearch/

et pour comprendre la difference entre query/match et query/term : https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-term-query.html