Documentation technique moteur de recherche Sirene

Sommaire

Prérequis

Pour utiliser cet outil, il faudra installer préalablement Elasticsearch (version 6.2.4), Kibana (version 6.2.4), Logstash et Composer.

La version d'Elasticsearch est importante car la syntaxe des requêtes change entre les versions et donc entraînera des erreurs en cas de version antérieure et peut-être supérieure.

Envoie des données à Elasticsearch

A. Formater les données

1. Mapping

Avant l'envoie des données grâce à Logstash, on va utiliser Kibana pour définir un "mapping" à Elasticsearch. Par défaut tout est un string. On veut que le champ "NOMEN_LONG" (nom de l'entreprise) soit utilisé pour une recherche prédictive, il faut donc le déclarer comme "completion" à Elasticsearch. Il faut impérativement faire cela avant même d'importer la moindre donnée. Cela correspond à la commande suivante (dans l'UI Kibana) :

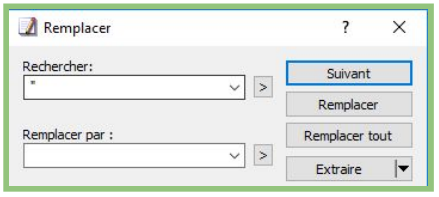
```
PUT b
{
  "mappings":{
    "_doc":{
      "properties":{
        "NOMEN_LONG":{
          "type":"completion"
        }
      }
    }
  }
}
```

2. Découpage des données

Une fois ce mapping fait, on ne pourra pas directement importer les données depuis le csv récupéré sur le site Sirene dans logstash à cause de sa structure (présence de double quotes). La solution trouvée, peu efficace mais fonctionnelle a été de couper le fichier de 8Go en fichiers de 1 000 000 de lignes soit entre 600 et 800 Mo. Pour faire cela, on peut utiliser CSVSplitter (en cochant la case “First row contains column header” afin de ne prendre que les données).

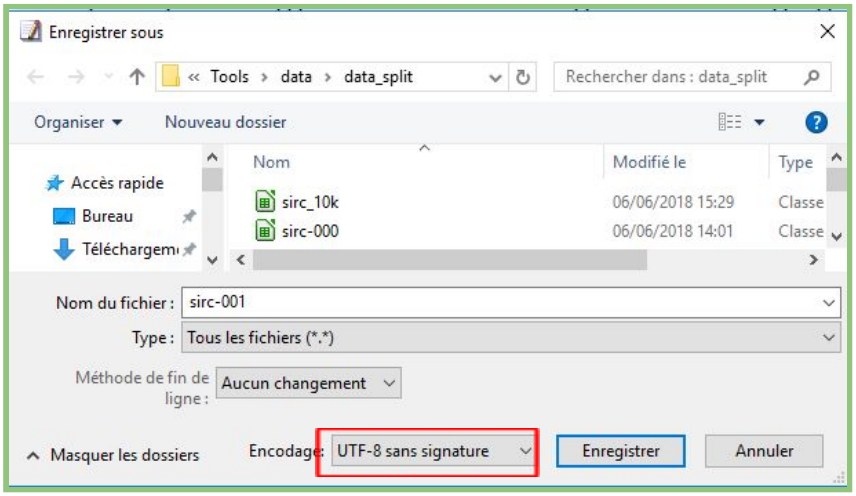
3. Reformatage

Une fois ces fichiers créés, on pourra les ouvrir avec EmEditor ou bien Nodepad ++ et “Rechercher-remplacer” tous les doubles quotes par rien.



4. Enregistrement utf-8

Une fois ces manipulations faites, il faudra enregistrer le fichier en veillant à choisir “Enregistrer sous” puis dans “Encodage” choisir “UTF-8 sans signature”.



B. Envoie des données

L'envoie des données se fait grâce à Logstash, avec le fichier "le_nom.config" ayant la structure suivante :

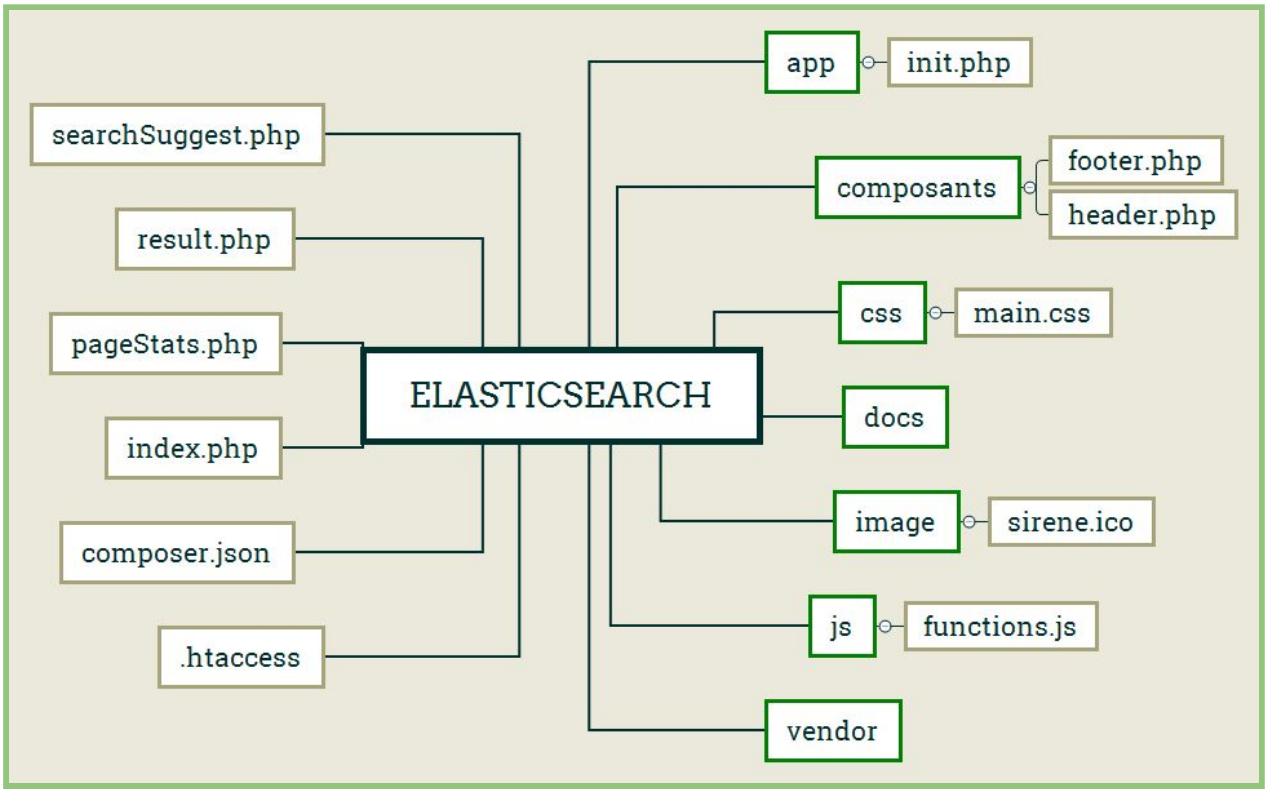
```
input{
	file{
		path => "adresse_du_csv\nom_csv.csv"
		codec => plain { charset => "UTF-8" }
		start_position => "beginning"
		sincedb_path => "/dev/null"
	}
}
filter{
	csv{
		separator => ";"
		columns=>[contient tous les champs de la base ex :  "champ1","champ2"....]
	}

	mutate {convert => ["LIBTEFET","integer"]} //permet de signifier que le champ est un entier

	mutate {convert => ["TCA","integer"]}
}
output{
	elasticsearch{
		hosts => "localhost:9200" //port est lancé Elasticsearch
		index => "b" //index des données
	}
	stdout{}
}
```

Une fois ce fichier créé, il suffira d'aller dans le dossier Logstash et d'y ouvrir une invite de commande. La commande permettant d'exécuter l'import des données est la suivante (sous windows, sous linux il faudra transformer les "\" en "/" :

```
logstash > bin\logstash -f C:\adresse_absolue_du_csv\nom_du_fichier_ logstash.config
```

Il faudra alors attendre que l'importation se fasse. Il sera possible de contrôler l'avancée de celle-ci sur Kibana avec la commande :
`GET b/_search?size=0` qui retournera un json ou "total" sera le nombre de données importées.

Architecture technique du site



A. Légende

Tous les documents n'ont pas été explicitement inclus, seuls les plus importants sont présents sur cette carte.

Les dossiers sont les encadrés verts et les fichiers les encadrés marrons.

Il faut ajouter à cette arborescence la présence d'un README pour plus de détails sur l'installations et de possibles complications.

B. Explications

1. Les dossiers

- **app** ⇒ contient le script de connexion à elasticsearch
- **composants** ⇒ contient le header et le footer
- **css** ⇒ contient les fichiers nécessaires au fonctionnement de Bootstrap ainsi qu'un fichier de style personnalisé ("main.css")
- **docs** ⇒ contient la structure siren utilisée, le livrable utilisateur et ce livrable
- **image** ⇒ contient le fichier .ico utilisé pour la favicon de l'outil
- **js** ⇒ contient les fichiers nécessaires au fonctionnement de Bootstrap, jquery et un fichier de scripts personnalisé ("functions.js")
- **vendor** ⇒ contient les fichiers générés par Composer

2. Les fichiers

- **index** ⇒ page de recherche, contient le formulaire de recherche générale et celui de recherche personnalisée
- **pageStats** ⇒ page qui affichera les statistiques choisies à partir de Kibana, pour l'instant page de tests avec des statistiques affichées
- **result** ⇒ page qui affiche les informations relatives à une entreprise lorsque l'utilisateur a cliqué sur celle-ci
- **searchSuggest** ⇒ page utilisée pour la recherche prédictive, appelée dans "index.php"
- **composer.json** ⇒ fichier utilisé pour exécuter Composer et installer les dépendances. Il faut réexécuter ce script lors de l'implémentation de l'outil au lieu de copier-coller le dossier "vendor".

Structure **composer.json** :

```
{
  "require": {
    "elasticsearch/elasticsearch": "~6.0"
  }
}
```

Commande utilisée : `> composer install` .

Point Elasticsearch

A. Structure des données

Elasticsearch fonctionne par index et type. Par exemple si on stocke des films on aurait une arborescence de ce genre : /movies/marvel/1 qui correspond au film dont l'id = 1, de type "marvel", dans l'index "movies".

Le "mapping" sera donc affecté à un type et un index précis. Le but va être de donner le format dans lequel nous voulons que certains champ soit implémentés. Chaque "document" (donnée Elasticsearch) sera constitué de la même façon, il aura tous les champs déclarés dans le mapping et à chacun correspondra une valeur.

Exemple d'un compte utilisateur ayant nom, prénom et âge en champ :

```
{
        "nom":"Durant",
        "prenom":"Pierre",
        "age":25
}
```

L'âge n'est pas entre quote car c'est un int.

B. Les requêtes

Les requêtes Elasticsearch peuvent être effectuées sur un champ en particulier, sur un ensemble de champs ou bien sur tous les champs. Les requêtes retournent une liste de résultats (10 maximum par défaut) au format json.

C. Un exemple

On souhaite récupérer l'entreprise ayant 019805324 comme numéro siren. Dans Kibana il suffit de saisir : GET b/_search?q=019805324

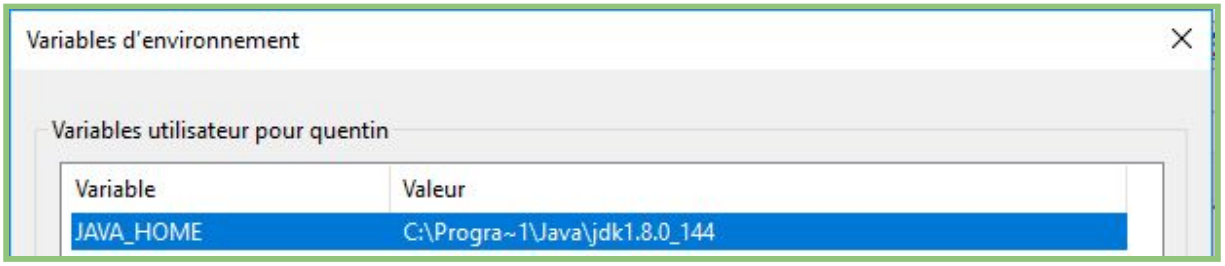
Le résultat de cette requête sera :

```
{
 "took": 27,       ⇒ temps d'exécution
 "timed_out": false, ⇒ erreur
 "_shards": {
  "total": 5,
  "successful": 5,
  "skipped": 0,
  "failed": 0
 },
 "hits": {
  "total": 1,      ⇒ nombre de résultats
  "max_score": 7.2375383, ⇒ meilleure pertinence
  "hits": [
   {
    "_index": "b", ⇒ index ou se trouve le résultat
    "_type": "doc", ⇒ type du résultat
    "_id": "NDdg2WMBaq-b25rFWy2f",
    "_score": 7.2375383, ⇒ pertinence du résultat
    "_source": {
     "TU": "5", ⇒ “champ”:”valeur”
     "IndRep": null, ⇒ //://
     "SIGLE": null, => //://
     "NICSIEGE": "00019", ⇒ //://
     [...]
     "SIREN":"019805324" ⇒ //://
     [...]
    }
   }
}
```

Annexe

A. Variables d'environnement

Lors de du premier lancement d'Elasticsearch il est possible d'avoir une erreur de type : "could not find java; set JAVA_HOME or ensure java is in PATH" cela est dû à l'absence ou à l'invalidité de la variable d'environnement JAVA_HOME. Pour changer cela, il faut aller modifier les variables d'environnement afin de modifier ou d'ajouter la variable JAVA_HOME avec l'adresse absolue du dossier java.
Exemple :



B. Config Elasticsearch

Il est possible de changer l'hôte Elasticsearch, pour cela il faudra modifier le fichier /Elasticsearch/config/elasticsearch.yml, on pourra alors modifier "network.host:xxx" en mettant l'adresse souhaitée. Il faudra dans ce cas modifier les fichiers faisant appel à Elasticsearch.

C. Config Kibana

Il est possible de changer l'hôte Kibana, pour cela il faudra modifier le fichier /Kibana/config/kibana.yml, on pourra alors modifier "server.host:xxx" afin de pouvoir accéder à kibana à l'adresse du server choisi au port 5601.