
# Guide pratique pour Elasticsearch

Travaux Pratiques

**ESILV**
nicolas.travers (at) devinci.fr

# Chapitre

# Table des matières

# Chapitre 1

# Mise en place

Le logiciel `elasticsearch`, développé en Java, est très facile d'installation et de déploiement dans un environnement distribué. Son but est d'intégrer du contenu semi-structuré (JSon) orienté texte et de permettre son interrogation.

L'API est implémentée sous forme de service REST que l'on peut interroger via le port 9200.

## 1.1 Installation Docker

Une image Docker utile pour tester elasticsearch est "nshou/elasticsearch-kibana".

```
docker pull nshou/elasticsearch-kibana
```

Ensuite créer votre container (possible avec le docker desktop) avec les redirections de ports nécessaires :

```
docker run -d -p 9200:9200 -p 5601:5601 --name elasticsearch-kibana nshou/elasticsearch-kibana
```

Le démarrage du serveur peut prendre un peu de temps (1 bonne minute), pour vérifier, aller sur l'URL : `http://localhost:9200`

Il fonctionnera dès que vous obtenez une réponse obtenue sous forme de document JSON :

```
{
        "name" : "f35479187df9",
        "cluster_name" : "elasticsearch",
        "cluster_uuid" : "-KYIrXJcTJ-ai_wI8vl8Eg",
        "version" : {
                "number" : "7.16.2",
                "build_flavor" : "default",
                "build_type" : "tar",
                "build_hash" : "2b937c44140b6559905130a8650c64dbd0879cfb",
                "build_date" : "2021-12-18T19:42:46.604893745Z",
                "build_snapshot" : false,
                "lucene_version" : "8.10.1",
                "minimum_wire_compatibility_version" : "6.8.0",
                "minimum_index_compatibility_version" : "6.0.0-beta1"
        },
        "tagline" : "You Know, for Search"
}
```

Ensuite, vous pouvez tester Kibana : `http://localhost:5601`. Dès que la page est lancée, le serveur est prêt à être utilisé.

## 1.2 Téléchargement et installation

Vous pouvez télécharger la dernière version sur :

— Elasticsearch : `https://www.elastic.co/fr/downloads/elasticsearch`

— Kibana : `https://www.elastic.co/fr/downloads/kibana`

Dans le répertoire de *elasticsearch*, dénommé ici par $ELASTIC, vous pourrez retrouver les répertoires suivants après décompression :

— `$ELASTIC/bin` : exécutables,

— `$ELASTIC/config` : fichiers de configuration,

— `$ELASTIC/plugins` : extensions,

— `$ELASTIC/logs` : fichiers de journalisation en cas de problèmes.

### 1.2.1 Configuration

Pour configurer un serveur, ouvrez le fichier suivant : `$ELASTIC/config/elasticsearch.yml`

— `cluster.name` : nom du cluster pour l'ensemble des noeuds *elastic*,

— `node.name` : nom du noeud que vous souhaitez démarrer (doit être unique pour un cluster),

— `index.number_of_shards` : nombre de serveurs (défaut 1),

— `index.number_of_replicas` : nombre de serveurs de réplication pour la tolérance aux pannes (défaut 0).

Pour faciliter l'administration, installez l'interface Web '*head*' dans une console.

```
$ELASTIC/bin/plugin -install mobz/elasticsearch-head
```

### 1.2.2 Lancer le serveur

*1.2.1* Dans une console, lancez le serveur : `$ELASTIC/bin/elasticsearch`
Ne pas éteindre ce serveur, ni fermer la fenêtre !

*1.2.2* Pour vérifier l'état, ouvrir dans un navigateur : `http://localhost:9200/?pretty` (ou utilisez *curl*[1])

*1.2.3* Pour éteindre le serveur : `curl -XPOST 'http://localhost:9200/_shutdown'`

*1.2.4* Pour ouvrir l'interface 'head' d'administration : `http://localhost:9200/_plugin/head`

## 1.3 Fonctionnement d'une base

Une base 'elastic' se nomme un **index**, vous pouvez comparer cela à une collection en NoSQL ou une table étendue en relationnel.

Ainsi, pour importer des données dans *elasticsearch*, il vous faut préciser cet index et ce type en prefixant chaque document importé par :

```
{"index":{"_index": "MA BASE","_id":1}}
```

L'identifiant "_id" doit être unique dans le type et sera associé au document suivant.

1. `https://curl.haxx.se/dlwiz/?type=bin`

# Chapitre 2

# Importation de données

Pour importer des données dans Elasticsearch, il y a trois manières de le faire :

— Service `bulk` avec curl (rapide),
— Interface web avec Kibana (lent mais facile),
— Logstash/Intégrations (automatisation)

## 2.1 Service bulk avec Curl

Cette manière d'importer les données utilise un exécutable (en ligne de commande) permettant de simuler les interactions avec des URLs (dont les API REST). Très utile pour faire des instructions simples.

Pour Windows : `https://curl.se/download.html#Win64`

Pour importer les données :

— Télécharger le dataset *movies_elastic*,
— Décompresser l'archive,
— Exécuter la commande

```
curl -XPUT localhost:9200/_bulk  -H "Content-Type: application/json" --data-binary @movies_elastic.json
```

— Ouvrir kibana : `http://localhost:5601`
— Aller sur : `Menu/Management/Stack Management`
`http://localhost:5601/app/management`
— Aller sur : `Data/Index Management`
`http://localhost:5601/app/management/data/index_management/indices`
Vérifier que l'**index** "*movies*" existe.
— Aller sur : `Kibana/Index Patterns`
`http://localhost:5601/app/management/kibana/indexPatterns`
— Chercher l'index "movies", et **ne pas** choisir de "timestamp field"
— C'est prêt !

## 2.2 Interface Web de Kibana

Vous pouvez importer votre dataset également en utilisant l'interface de Kibana :

— Aller sur kibana : `http://localhost:5601`
— Aller sur : `Menu/Kibana/Index Patterns`
`http://localhost:5601/app/management/kibana/indexPatterns`
— Choisir "*upload a file*"
— Téléverser (drag and drop) le fichier "movies_elastic.json"
— Cliquer sur "*Import*" en bas à gauche.
— Donner un nom à l'index "movies"
⚠Cette opération prend du temps car c'est une interface Web. Et oui, il faut préférer `curl` ou `intégrations` (comme logstash)

## 2.3 Intégrations

Chaque connecteur à une base de données ou système est proposé ici : `https://www.elastic.co/fr/integrations/data-integrations`

Le traditionnel est `logStash` : `https://www.elastic.co/fr/logstash/`

Son utilisation n'est pas traité dans ce guide.

# Chapitre 3

# Interrogation

Toute requête se fait sur l'API REST, elle doit contenir :

— le nom de l'index
— le nom du type
— Un opérateur de recherche : _count, _search
— Optionnel un identifiant

## 3.1 Principe

### 3.1.1 Avec `curl`

`curl -XGET 'http://localhost:9200/tests/test/1'`
Index 'tests', de type 'test', pour le premier document (_id=1)
`curl -XGET 'http://localhost:9200/tests/test/_count'`
Compte le nombre de documents de 'tests/test'

### 3.1.2 Avec `Kibana`

Aller dans l'interface Kibana sur : `Menu/Management/Dev Tools` `http://localhost:5601/app/dev_tools#/console`

## 3.2 Requête de 'Matching'

Le service `_search` permet de faire des requêtes simples

— Recherche de mots dans l'ensemble de chaque document

```
curl -XGET 'http://localhost:9200/tests/test/_search?q=alien&pretty=true'
```

— Dans le titre de chaque document

```
curl -XGET 'http://localhost:9200/tests/test/_search?q=title:alien&pretty=true'
```

— Combinaison de critères avec l'espace (%20 pour une URL). Ici, la négation est utilisée ("-")

```
curl -XGET 'http://localhost:9200/tests/test/_search?q=title:alien%20-year:1992&pretty=true'
```

— Pour avoir une requête complexe en utilisant le DSL (Domain Specific Language), il faut envoyer un document JSon "requête"

```
curl -XGET http://localhost:9200/tests/test/_search?pretty -d '
{
  "query" : ...
}'
```

— Requêtes de simple correspondance (matching)

```
{
  "query" : {
    "match" : { "nom_de_l'attribut" : "la\_valeur\_a\_chercher"}
  }
}
```

— Requêtes d'intervalles (range)

```
{
  "query" : {
    "range" : { "nom_de_l'attribut" : "la\_borne"}
  }
}
```

— Notions de critère optionnel 'should' (avec calcul de score de pertinence) et d'obligatoire 'must' (sans score). Il faut utiliser le critère en imbriquant dans un opérateur 'bool'

```
{"query":{
  "bool": {
    "should": { "match": { "nom_de_l'attribut" : "valeur" }},
    "must": { "range": { "nom_de_l'attribut" : "valeur"}},
    "must_not":[
        {"match":{"nom_de_l'attribut" : "valeur"}},
        {"match":{"nom_de_l'attribut" : "valeur"}},
        ...
    ]
}}}
```

— Une requête avec l'opérateur '*filter*' ne calcule aucun score pour la requête correspondante (même avec 'should')

```
{
  "query": {
    "filtered": {
      "query": {
        "match": {"nom_de_l'attribut" : "valeur"}
      },
      "filter": {
        "range": {"nom_de_l'attribut" : "valeur"}
}}}}
```

Vous pourrez trouver la syntaxe complète ici :
`https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html`

## 3.3 Requêtes d'agrégats

Il est possible de regrouper les documents pour pouvoir appliquer une agrégation à des fins de statistiques. Pour cela, il faut utiliser la clé "aggs" dans les documents requêtes, donner la clé où prendre les valeurs "terms : field : XXX" (identique à un GROUP BY) et donner la clé où mettre la valeur d'agrégation. Par défaut, cela va compter le nombre d'occurences.

```
{"aggs" : {
        "nom_de_la_cle_en_sortie" : {
                "terms" : {"field" : "nom_de_la_cle_a_grouper"}
}}}
```

Attention, cette méthode va faire décomposer les valeurs de la clé "nom_de_la_cle_a_grouper". Ainsi, un texte se verra décomposer en ensemble de mots pour être ensuite regroupés (par mot).

On peut également changer "`terms`" en "`avg`", "`min`", "`max`", "`cardinality`" (distinct)... Voire même ressortir les mots importants "`significant_terms`" (grâce à leurs occurences dans un ensemble filtrés)

Il est possible de filtrer les documents avant l'agrégat. Il suffit de faire une requête classique "`query`".

```
{"query" :{
                "match" : {"cle_a_filtrer" : "valeur"}
        }
 ,"aggs" : {
                "cle_de_sortie" : {
                        "avg" : {"field" : "cle_a_grouper"}
        }}}
```

Il est possible d'imbriquer les agrégats pour grouper par type :

```
{"aggs" : {
        "nom_groupe" : {
                "terms" : {
                        "field" : "cle_du_group_by"
                },
                "aggs" : {
                        "nom_sous_groupe" : {
                                "avg" : {"field" : "cle_pour_la_moyenne"}
                        }}
                }
        }}
```

Le tri fonctionne de la même manière avec la clé "`order`" en utilisant le nom de la clé à trier.
`"order" : { "cle_pour_la_moyenne" : "desc" }`

Il est également possible de grouper par plages de valeur plutot que les valeurs du document :

```
{"aggs" : {
        "nom_cle_sortie" : {
                "range" : {
                        "field" : "cle_ou_prendre_la_valeur",
                        "ranges" : [
                                {"to" : "valeur_max"},
                                {"from" : "valeur_min", "to" : "valeur_max"},
                                {"from" : "valeur_min"}
                        ]
                }
                }}}
```

## 3.4 INSERT

L'index et le type sont obligatoires. Chaque document inséré DOIT avoir la même structure que le premier document inséré (celui qui détermine le schéma du 'type').

```
curl -XPOST 'http://localhost:9200/tests/test2/' -d "{'test':1, 'title' : 'my new elastic document'}"
```