TP MOTEUR DE RECHERCHE – ELK

Utiliser la VM VirtualBox nommée `nosql2019`

- Sous LINUX, la récupérer sur le serveur `/softwares/sync/VMs/` (si vous n'avez pas assez de place localement, supprimer la VM `TP NOSQL 2018`)
- OS : Ubuntu 64 bits
- RAM : 8 Go
- Augmenter la taille de la mémoire vidéo au maximum + activer l'accélération 3D
- Login (root) : `nosql` ; mot de passe : `nosql`
- Activer le presse-papier partagé (bidirectionnel)
- Si l'affichage de la VM est en faible résolution, installer les Additions Invité (drivers) : Menu *Périphériques* puis *Insérer l'image CD des Additions Invité…* L'image d'un CD va alors être chargée et les drivers installés. Redémarrer si nécessaire la machine virtuelle et changer la résolution de l'affichage.

Comment lire ce TP ?

Ceci est une question à réaliser

```
Ceci est un morceau de code ou une commande Bash à exécuter
```

Ceci est un cadre contenant des informations utiles

Ceci est un cadre avec des informations importantes

INTRODUCTION

Le but de ce TP est de découvrir et prendre en main la suite ELK avec une installation rapide, une indexation de gros volumes de données et d'effectuer des recherche & statistiques sur cette base. Pour cela nous allons utiliser deux CSV exportés d'une base de données de séries TV (Base de données à jour au 01/01/2018, soit 16277 séries pour 862482 épisodes).

Le répertoire Elasticsearch du répertoire personnel *(/home/nosql/elasticsearch)* sera notre répertoire principal de travail.

Pour commencer, ouvrir le fichier install-es.sh se trouvant dans le dossier /home/nosql/elasticsearch. Il permet de télécharger et d'installer l'ensemble des logiciels (ES, Kibana, Logstash, etc.).

Vous pouvez modifier le nom du cluster Elasticsearch. Par exemple, cpe-tp au lieu de dim-tp :

```
#!/bin/bash
echo "######################"
echo "### Downloading ES ###"
echo "######################"

wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.5.4.tar.gz
tar -xvzf elasticsearch-6.5.4.tar.gz
mv elasticsearch-6.5.4 elasticsearch
rm -f elasticsearch-6.5.4.tar.gz

echo "############################"
echo "### Downloading Logstash ###"
echo "############################"

wget https://artifacts.elastic.co/downloads/logstash/logstash-6.5.4.tar.gz
tar -xvzf logstash-6.5.4.tar.gz
mv logstash-6.5.4 logstash
mkdir logstash/conf
touch logstash/conf/csvload.conf
rm -f logstash-6.5.4.tar.gz

echo "############################"
echo "### Downloading Kibana ###"
echo "############################"

wget https://artifacts.elastic.co/downloads/kibana/kibana-6.5.4-linux-x86_64.tar.gz
tar -xvzf kibana-6.5.4-linux-x86_64.tar.gz
mv kibana-6.5.4-linux-x86_64 kibana
rm -f kibana-6.5.4-linux-x86_64.tar.gz

echo "######################"
echo "### Configuring ES ###"
echo "######################"

sed -i -e 's/#cluster.name: my-application/cluster.name: cpe-tp/g' elasticsearch/config/elasticsearch.yml
echo "OK"

echo "############################"
echo "### Downloading Cerebro ###"
echo "############################"

wget "https://github.com/lmenezes/cerebro/releases/download/v0.8.1/cerebro-0.8.1.zip"
unzip cerebro-0.8.1.zip
mv cerebro-0.8.1 cerebro
rm -f cerebro-0.8.1.zip
echo "OK"

echo "Install successful !"
```

Lancer via le terminal le script :

```
sh install-es.sh
```

1. INGESTION DES DONNEES

Logstash va être notre ETL pour lire, transformer et envoyer nos données CSV vers Elasticsearch. Ayez toujours un onglet ouvert sur la documentation de Logstash : Https://www.elastic.co/guide/en/logstash/current/index.html

Un fichier de configuration dans le dossier `conf` de Logstash a été créé (`logstash/conf/csvload.conf`). C'est dans ce dossier que doivent être stockés les différents fichiers de configuration. Ces fichiers sont vérifiés toutes les 3s pour intégrer les modifications apportées. N'oubliez donc pas de sauvegarder.

Pour lancer Logstash avec un fichier de configuration, il suffit d'exécuter cette commande depuis `/elasticsearch` :

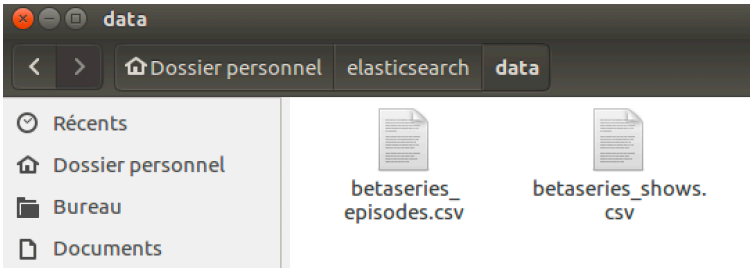
```
logstash/bin/logstash -f logstash/conf/<VotreFichierDeConf>
```

Si vous exécutez la commande, vous obtiendrez, pour l'instant, des erreurs, puisque le contenu du fichier conf est vide.

Logstash met beaucoup de temps à démarrer, d'autant plus dans une VM. Soyez patient !

Pour arrêter Logstash, un simple Ctrl+C suffit pour lui envoyer un signal d'extinction

N'hésitez pas à ouvrir les deux fichiers CSV du dossier `Data` et regarder leur contenu.



a) Première configuration composée d'un input et d'un output qui lisent les fichiers contenus dans le répertoire `elasticsearch/data` et les affichent sur le terminal. Modifier le contenu du fichier `csvload.conf`.

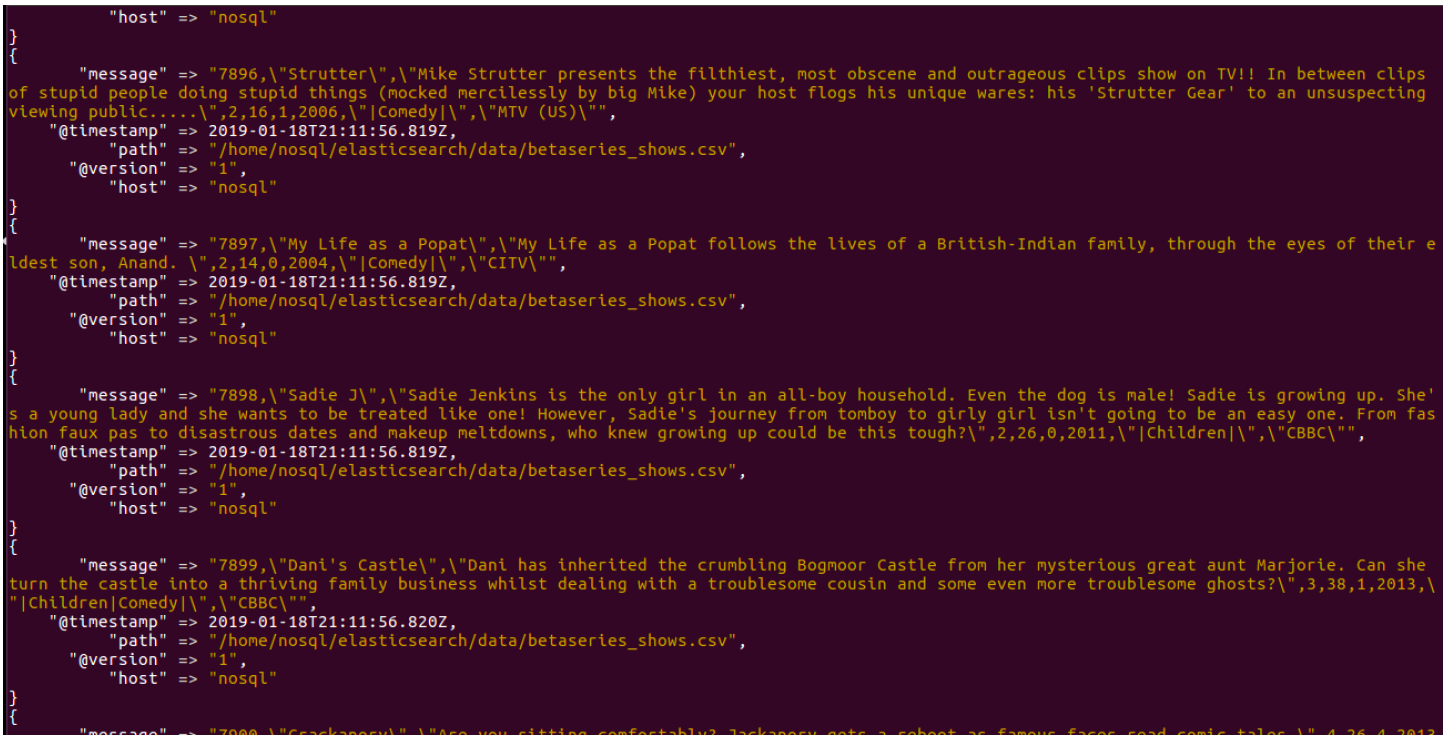
Code :

```
# Fichier de configuration logstash
input {
    file {
    # chemin d'accés
        path => "/home/nosql/elasticsearch/data/*"
    # curseur de lecture à supprimer pour relancer
        sincedb_path => "/dev/null"
    # Lecture depuis le début
        start_position => "beginning"
    }
}

filter{

}
output {
    # affichage sur la sortie standard
    stdout {
        codec => "rubydebug"
    }
}
```

Explications :

- Nous utilisons les plugins `file` en input et `stdout` en output (avec un codec `rubydebug` pour plus de lisibilité) :
  - https://www.elastic.co/guide/en/logstash/current/plugins-inputs-file.html
  - https://www.elastic.co/guide/en/logstash/current/plugins-outputs-stdout.html
- * : pour sélectionner les deux fichiers en même temps dans le path
- Le plugin `file` commence à lire un fichier par la fin par défaut. Ici, nous lui indiquons de commencer par le début (`start_position => "beginning"`).
- Le plugin `file` comporte un paramètre `sincedb` qui permet de garder une trace de son exécution. Il faudra soit supprimer le fichier `.since_db` à chaque fois soit le rediriger vers `/dev/null`, comme nous l'avons fait.
- Pour le moment, aucune transformation n'est réalisée (plugin `filter` : https://www.elastic.co/guide/en/logstash/current/core-operations.html)

Résultat (après avoir relancé Logstash) **:** Chaque ligne des fichiers s'affiche dans le terminal



Vous pouvez utiliser CTRL+C pour arrêter l'exécution.

On remarque les champs :

- Message : contient toutes les données concaténées que l'on souhaite récupérer
- Path : chemin ver le fichier contenant la ligne
- @timestamp : date actuelle
- Etc.

b) Maintenant, on va faire comprendre à Logstash que le fichier est un CSV à interpréter. Rajouter dans `filter` un plugin `csv` qui va parser nos données. La première ligne du CSV contient le nom des colonnes. Il faut ensuite supprimer le champ message qui contient la ligne CSV et ne sera pas utile.

> ***Indications :***
>
> ***- Vu qu'on a deux formats différents de csv, il faut utiliser IF / ELSE (https://www.elastic.co/guide/en/logstash/current/event-dependent-configuration.html) avec éventuellement IN (pour utiliser le champ, il faudra le mettre entre [], exemple : IF [monchamp]=="valeur") et donner manuellement les colonnes du CSV à la configuration des deux plugins CSV (https://www.elastic.co/guide/en/logstash/current/plugins-filters-csv.html) .***

- Les colonnes sont fournies en annexes.

- Le plugin `mutate` permet de modifier un objet (ajout/suppression de champ, modification de valeur, …). Utiliser l'option `remove_field` : https://www.elastic.co/guide/en/logstash/current/plugins-filters-mutate.html.

Exemple de code pour la transformation des données des séries :

```
input {
    file {
        path => "/home/nosql/elasticsearch/data/betaseries_shows.csv"
        sincedb_path => "/dev/null"
        start_position => "beginning"
    }
}
filter{

# on parse le fichier shows
        csv {
            columns => [
"show_id","title","description","seasons","episodes","follow_count","creation_year","genres","network"
]
            skip_empty_rows => true
            separator => ","
        }
}

output {
    stdout {
        codec => "rubydebug"
    }
}
```

Résultats : Le résultat en stdout s'affiche maintenant découpé en champs du nom des colonnes du CSV dans un format attendu (JSON, RUBY, …).

```
        "@version" => "1",
     "description" => "Série qui retrace des faits juridiques réels datant du 18ème siècle. Ces histoires s'inspirent de la vie de l'avocat Wi
lliam Garrow, qui a introduit le 'innocent jusqu'à preuve du contraire' au Old Bailey de Londres.",
         "seasons" => "3",
         "show_id" => "2416",
            "host" => "nosql",
    "follow_count" => "65",
         "network" => "BBC One",
        "episodes" => "12"
}
{
    "creation_year" => "2013",
         "message" => "7034,\"Susanna\",\"Une femme très ambitieuse professionnellement doit mettre entre parenthèse son activité pour s'occup
er de la fille de sa sœur souffrant de problèmes mentaux.\",1,12,9,2013,\"|Drama|Family|\",\"YouTube\"",
          "genres" => "|Drama|Family|",
       "@timestamp" => 2019-01-18T21:46:42.240Z,
           "title" => "Susanna",
            "path" => "/home/nosql/elasticsearch/data/betaseries_shows.csv",
        "@version" => "1",
     "description" => "Une femme très ambitieuse professionnellement doit mettre entre parenthèse son activité pour s'occuper de la fille de s
a sœur souffrant de problèmes mentaux.",
         "seasons" => "1",
         "show_id" => "7034",
            "host" => "nosql",
    "follow_count" => "9",
         "network" => "YouTube",
        "episodes" => "12"
}
{
    "creation_year" => "2011",
         "message" => "7035,\"Numberphile\",\"Videos about numbers - it's that simple. Videos by Brady Haran\",5,239,17,2011,\"|Documentary|\"
,\"YouTube\"",
          "genres" => "|Documentary|",
       "@timestamp" => 2019-01-18T21:46:42.240Z,
           "title" => "Numberphile",
            "path" => "/home/nosql/elasticsearch/data/betaseries_shows.csv",
        "@version" => "1",
     "description" => "Videos about numbers - it's that simple. Videos by Brady Haran",
         "seasons" => "5",
         "show_id" => "7035",
            "host" => "nosql",
    "follow_count" => "17",
         "network" => "YouTube",
        "episodes" => "239"
}
```


En fonction des indications précédentes et du code fourni, intégrer également la transformation du second fichier. Supprimer le champ message pour chaque fichier.

Résultats attendus :

- Séries :

```
     "follow_count" => "65",
          "seasons" => "3",
             "path" => "/home/nosql/elasticsearch/data/betaseries_shows.csv",
          "show_id" => "2416",
         "@version" => "1",
      "description" => "Série qui retrace des faits juridiques réels datant du 18ème siècle. Ces histoires s'inspirent de la vie de l'avocat Wi
lliam Garrow, qui a introduit le 'innocent jusqu'à preuve du contraire' au Old Bailey de Londres.",
       "@timestamp" => 2019-01-18T22:01:35.038Z,
           "title" => "Garrow's Law - Tales from the Old Bailey",
    "creation_year" => "2009",
          "network" => "BBC One",
             "host" => "nosql",
           "genres" => "|Drama|"
}
{
         "episodes" => "12",
     "follow_count" => "9",
          "seasons" => "1",
             "path" => "/home/nosql/elasticsearch/data/betaseries_shows.csv",
          "show_id" => "7034",
         "@version" => "1",
      "description" => "Une femme très ambitieuse professionnellement doit mettre entre parenthèse son activité pour s'occuper de la fille de s
a sœur souffrant de problèmes mentaux.",
       "@timestamp" => 2019-01-18T22:01:35.038Z,
           "title" => "Susanna",
    "creation_year" => "2013",
          "network" => "YouTube",
             "host" => "nosql",
           "genres" => "|Drama|Family|"
}
{
         "episodes" => "239",
     "follow_count" => "17",
          "seasons" => "5",
             "path" => "/home/nosql/elasticsearch/data/betaseries_shows.csv",
          "show_id" => "7035",
         "@version" => "1",
      "description" => "Videos about numbers - it's that simple. Videos by Brady Haran",
       "@timestamp" => 2019-01-18T22:01:35.038Z,
           "title" => "Numberphile",
    "creation_year" => "2011",
          "network" => "YouTube",
             "host" => "nosql",
           "genres" => "|Documentary|"
}
```

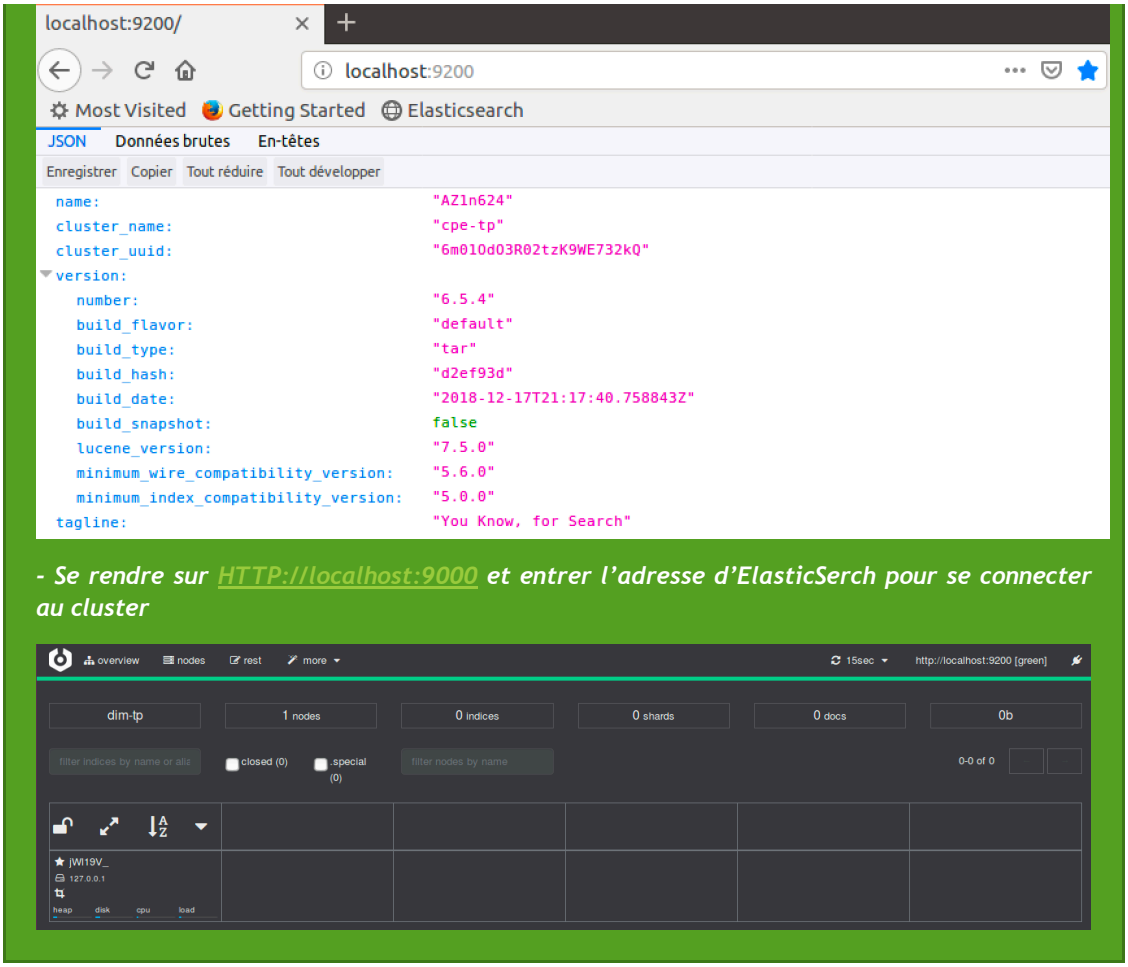

- Episodes :

```
}
{
              "@version" => "1",
            "@timestamp" => 2019-01-18T22:04:26.616Z,
    "released_timestamp" => "1147212000",
           "description" => "Jack, Sawyer, John et Kate retournent à la trappe pour empêcher Ana Lucia de commettre l'irréparable. Sur place, i
ls découvrent Michael blessé. Le prisonnier s'est échappé laissant derrière lui les cadavres de Ana Lucia et Libby... Mr Eko entraîne Locke à l
a poursuivre du fuyard. Sa véritable motivation est d'aider John à découvrir le mystère du point d'interrogation sur l'île...",
                "code" => "S02E21",
             "episode" => "21",
                "host" => "nosql",
          "show_title" => "Lost",
          "episode_id" => "167077",
               "title" => "?",
                "path" => "/home/nosql/elasticsearch/data/betaseries_episodes.csv",
             "show_id" => "6",
              "season" => "2"
}
{
              "@version" => "1",
            "@timestamp" => 2019-01-18T22:04:26.616Z,
    "released_timestamp" => "1147816800",
           "description" => "Michael tente de convaincre Jack et quelques autres de l'aider à sauver Walt. Le médecin s'apprête à organiser une
 armée pour traverser l'île, mais Michael refuse de prendre des risques inutiles. Il tient à ce qu'ils soient en petit comité... A la grande su
rprise de Charlie, Eko s'installe dans la trappe, oubliant la construction de l'église... ",
                "code" => "S02E22",
             "episode" => "22",
                "host" => "nosql",
          "show_title" => "Lost",
          "episode_id" => "167078",
               "title" => "Three Minutes",
                "path" => "/home/nosql/elasticsearch/data/betaseries_episodes.csv",
             "show_id" => "6",
              "season" => "2"
}
{
              "@version" => "1",
            "@timestamp" => 2019-01-18T22:04:26.616Z,
    "released_timestamp" => "1148421600",
           "description" => "Alors qu'ils font leurs adieux à Ana Lucia et Libby, les disparus découvrent un bateau qui se dirige vers eux. A b
ord se trouve Desmond ! A défaut de ne pouvoir fuir l'île, celui-ci se noie dans l'alcool... Sayid, persuadé de la trahison de Michael, propose
 à Jack un plan pour surprendre les Autres... Quant à Locke, agacé d'avoir été l'instrument d'une expérience durant tout ce temps, il veut savo
ir ce qu'il se passe lorsqu'on n'appuie plus sur le bouton...",
                "code" => "S02E23",
             "episode" => "23",
                "host" => "nosql",
          "show_title" => "Lost",
          "episode_id" => "167079",
               "title" => "Live Together, Die Alone (1)",
                "path" => "/home/nosql/elasticsearch/data/betaseries_episodes.csv",
             "show_id" => "6",
              "season" => "2"
}
```


C'est le moment de démarrer Elasticsearch et Cerebro, un outil de requête et monitoring pour Elasticsearch. Il suffit d'exécuter le script `./start.sh` (dans un autre terminal) pour les démarrer tous les deux.

```
nosql@nosql:~$ cd elasticsearch/
nosql@nosql:~/elasticsearch$ ./start.sh
##############################
### Launching ES  & Cerebro ###
##############################
nosql@nosql:~/elasticsearch$ [info] play.api.Play - Application started (Prod)
[info] p.c.s.AkkaHttpServer - Listening for HTTP on /0:0:0:0:0:0:0:0:9000
```

Pour vérifier que tout a démarré :
- Se rendre sur HTTP://localhost:9200 -> C'est l'adresse d'ElasticSearch. Vérifier que le nom du cluster est bien « cpe-tp »

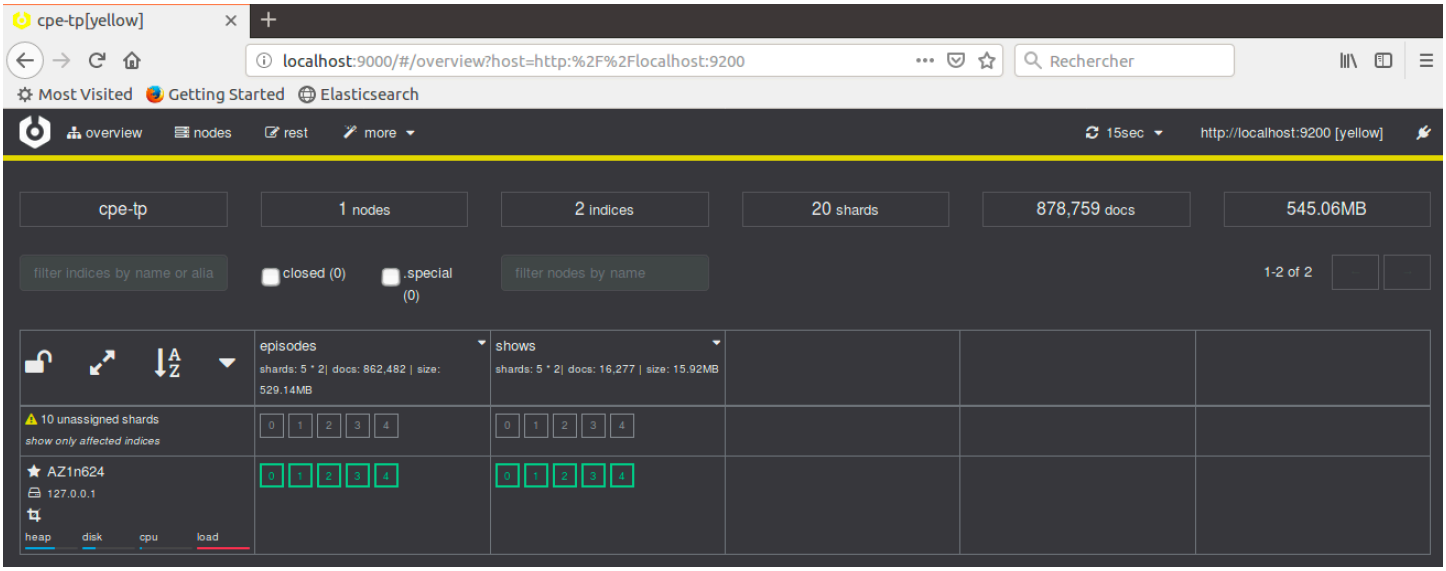


- Se rendre sur HTTP://localhost:9000 et entrer l'adresse d'ElasticSerch pour se connecter au cluster

c) Il est maintenant temps d'envoyer nos données formatées sur Elasticsearch. Nous allons donc faire deux index (indices) : shows et episodes. Remplacer le plugin stdout par un plugin `elasticsearch`. Vu que Logstash traite nos fichiers CSV en même temps, il faut qu'on différencie les sorties : épisodes vers un index « episodes », séries vers un index « shows » (utiliser `index` dans le plugin `elasticsearch`). Comme précédemment, utiliser un if pour cela.

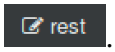
Réutiliser le même IF que pour le plugin CSV mais en mettant une des sorties elasticsearch à la place

Résultat attendu : Sur Cerebro, il y a maintenant deux index visibles (*shows et episodes*) avec des documents à l'intérieur (16 277 séries et 862 482 épisodes) sur la partie « overview ». Il faudra attendre un peu, le temps de charger tous les épisodes…



Maintenant que toutes les données sont dans Elasticsearch, vous pouvez fermer Logstash et ses dossiers.

II. QUERY ELASTICSEARCH

Cerebro permet d'exécuter des requêtes REST directement sur Elasticsearch dans l'onglet « rest » 

.

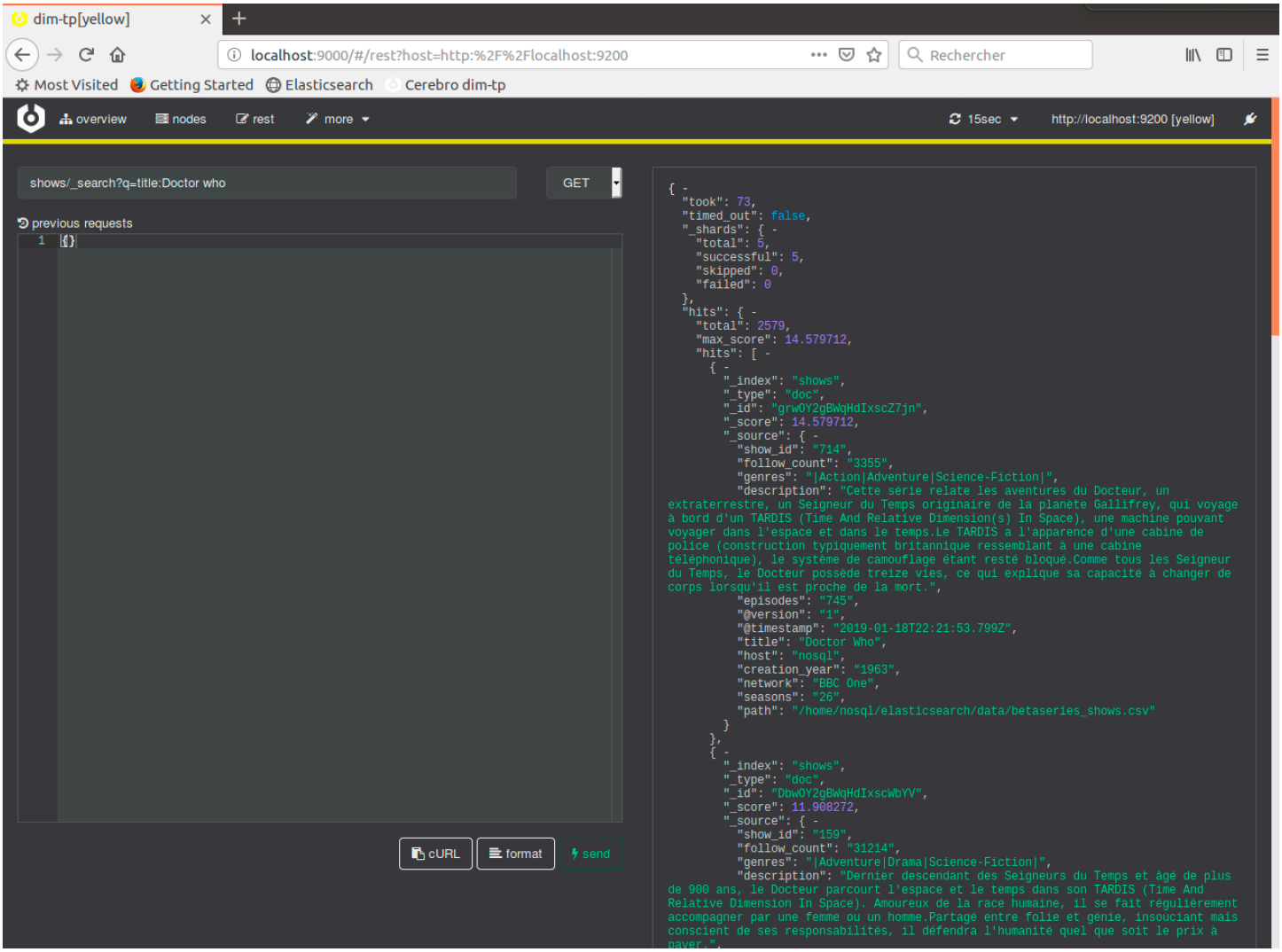
Les requêtes Elasticsearch s'effectuent en HTTP sur des *Endpoint* REST qui expriment l'index cible et l'action à réaliser (CRUD). Autrement dit, on effectue de simples requêtes HTTP sur un chemin particulier avec la recherche en paramètre ou dans le corps de celle-ci.

> ***Rappel sur le REST HTTP et un CRUD :***
> ***POST -> Création d'objet dans le corps de l'appel (CREATE)***
> ***GET -> Affiche (READ)***
> ***PUT -> mise à jour (UPDATE)***
> ***DELETE -> Suppression (DELETE)***
> ***plus de détails ici : Https://blog.nicolashachet.com/niveaux/confirme/larchitecture-rest-expliquee-en-5-regles/***

Les chemins Elasticearch se décomposent de cette façon : `{index}/{type}/{id}` ou `{index}/{type}/{action}` (Rappel : `{type}` n'est pas obligatoire). L'action pour faire des recherches est `_search`. Ainsi pour rechercher dans les épisodes, le chemin sera `episodes/_search`

Il existe deux systèmes de requêtes sur Elasticsearch : la *Query DSL* et la *Query String*. On va commencer par cette dernière.

Une query string s'effectue en GET sur *shows/_search* avec en paramètre `q=MA_QUERY`. **Exemple :** *GET shows/_search?q=title:Doctor who*. Dans l'onglet « rest », on obtient 2579 enregistrements :



Pour voir le format de ces requêtes, vous pouvez vous reporter au cours ou à la documentation d'Elasticsearch (Https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-query-string-query.html#query-string-syntax)

a) Réaliser ces requêtes :

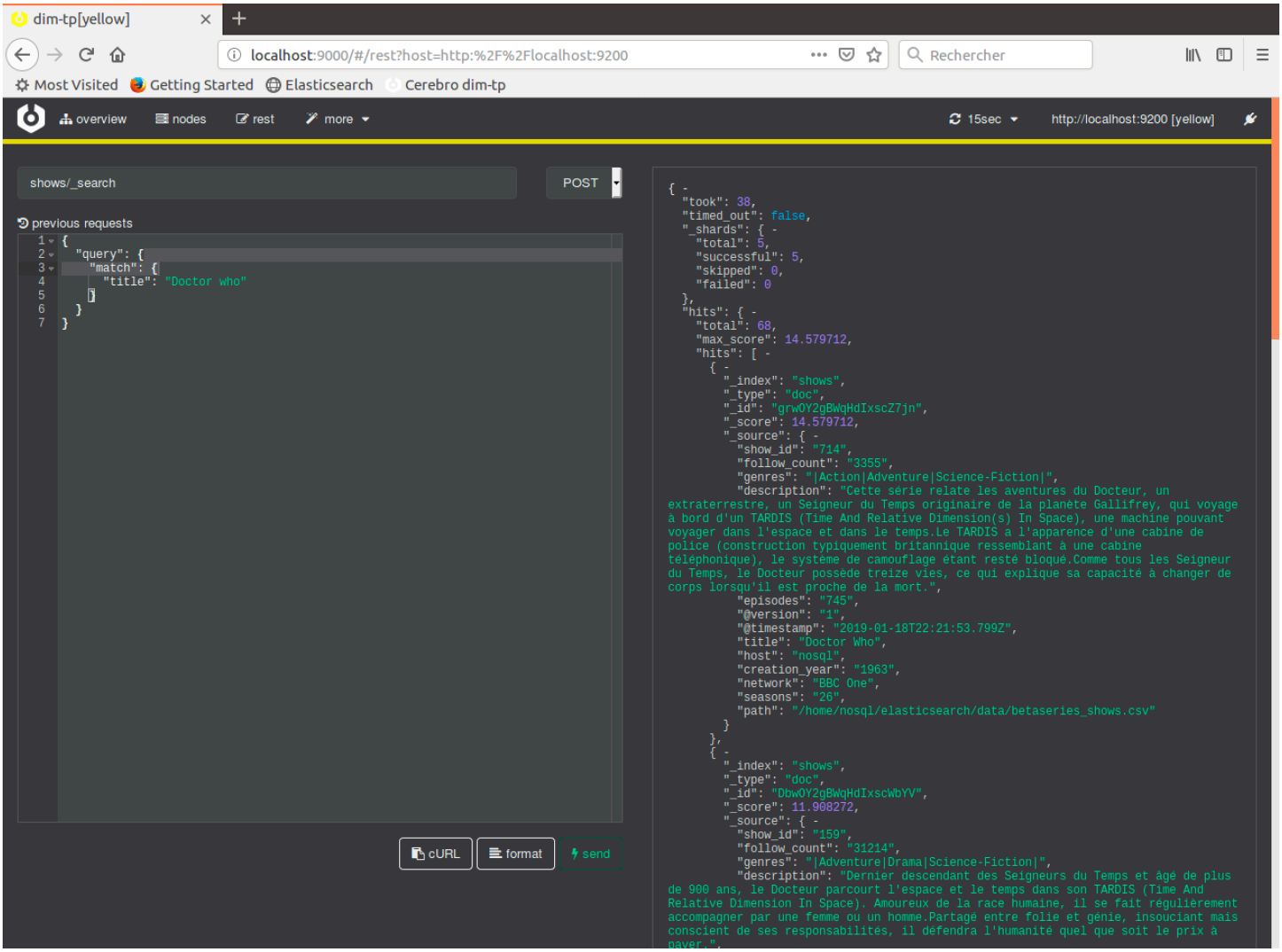
- Les séries qui ont dans le titre le mot « detective » (afficher tous les résultats) -> 37 résultats
- Les séries qui ont dans le titre le mot « detective » et qui ont plus de 3 saisons -> 4 résultats
- Les séries qui ont dans le titre le mot « batman » mais sans le mot pingouin dans la description -> 9 résultats
- Les Séries qui ont dans la description les mots « alien » ou « robot » du genre « Comedy » et qui ont été créés dans les années 90 -> 5 résultats

Nous allons maintenant passer au query DSL.

La Query DSL s'effectue sur le même chemin que les query string mais sans aucun paramètre. Toute la requête sera contenue dans un JSON dans le corps de l'appel HTTP qui est envoyé via un POST.

Le AND et OR deviennent une bool query et les échelles deviennent des range query. Le chapitre de la documentation sur la recherche full text vous donne toutes les clés pour comprendre le système de query DSL (Https://www.elastic.co/guide/en/elasticsearch/guide/current/full-text-search.html)

Exemple : recherche exacte sur le titre "Doctor who"



b) Essayer maintenant de faire les requêtes précédentes en Query DSL.

c) Essayer de faire une recherche sur les séries Start Trek en triant par nombres d'épisodes (champ episodes dans les documents shows

ATTENTION : cela devrait normalement vous indiquer une erreur : les « fielddata » ne sont pas activés pour ce champ texte (Cf. écran suivant)

```
{ -
  "error": { -
    "root_cause": [ -
      { -
        "type": "illegal_argument_exception",
        "reason": "Fielddata is disabled on text fields by default. Set
fielddata=true on [episodes] in order to load fielddata in memory by uninverting the
inverted index. Note that this can however use significant memory. Alternatively use
a keyword field instead."
      }
    ],
    "type": "search_phase_execution_exception",
    "reason": "all shards failed",
    "phase": "query",
    "grouped": true,
    "failed_shards": [ -
      { -
        "shard": 0,
        "index": "shows",
        "node": "jW119V_KT0a0aJh_neB6Kg",
        "reason": { -
          "type": "illegal_argument_exception",
          "reason": "Fielddata is disabled on text fields by default. Set
fielddata=true on [episodes] in order to load fielddata in memory by uninverting the
inverted index. Note that this can however use significant memory. Alternatively use
a keyword field instead."
        }
      }
    ],
    "caused_by": { -
      "type": "illegal_argument_exception",
      "reason": "Fielddata is disabled on text fields by default. Set fielddata=true
on [episodes] in order to load fielddata in memory by uninverting the inverted
index. Note that this can however use significant memory. Alternatively use a
keyword field instead.",
      "caused_by": { -
        "type": "illegal_argument_exception",
        "reason": "Fielddata is disabled on text fields by default. Set
fielddata=true on [episodes] in order to load fielddata in memory by uninverting the
inverted index. Note that this can however use significant memory. Alternatively use
a keyword field instead."
      }
    }
  },
  "status": 400
}
```


Vérifier le mapping de l'index pour voir le type du champ « episodes » (GET _all/_mapping)

> ***Lorsque que Logstash indexe les données sur Elasticsearch, il applique un mapping par défaut qui considère tous les champs comme des champs texte avec un sous-champs analysé. On peut définir un mapping personnalisé dans le plugin output de Logstash mais nous allons plutôt faire une réindexation directement dans Elasticsearch.***

III. Reindexation et Mapping

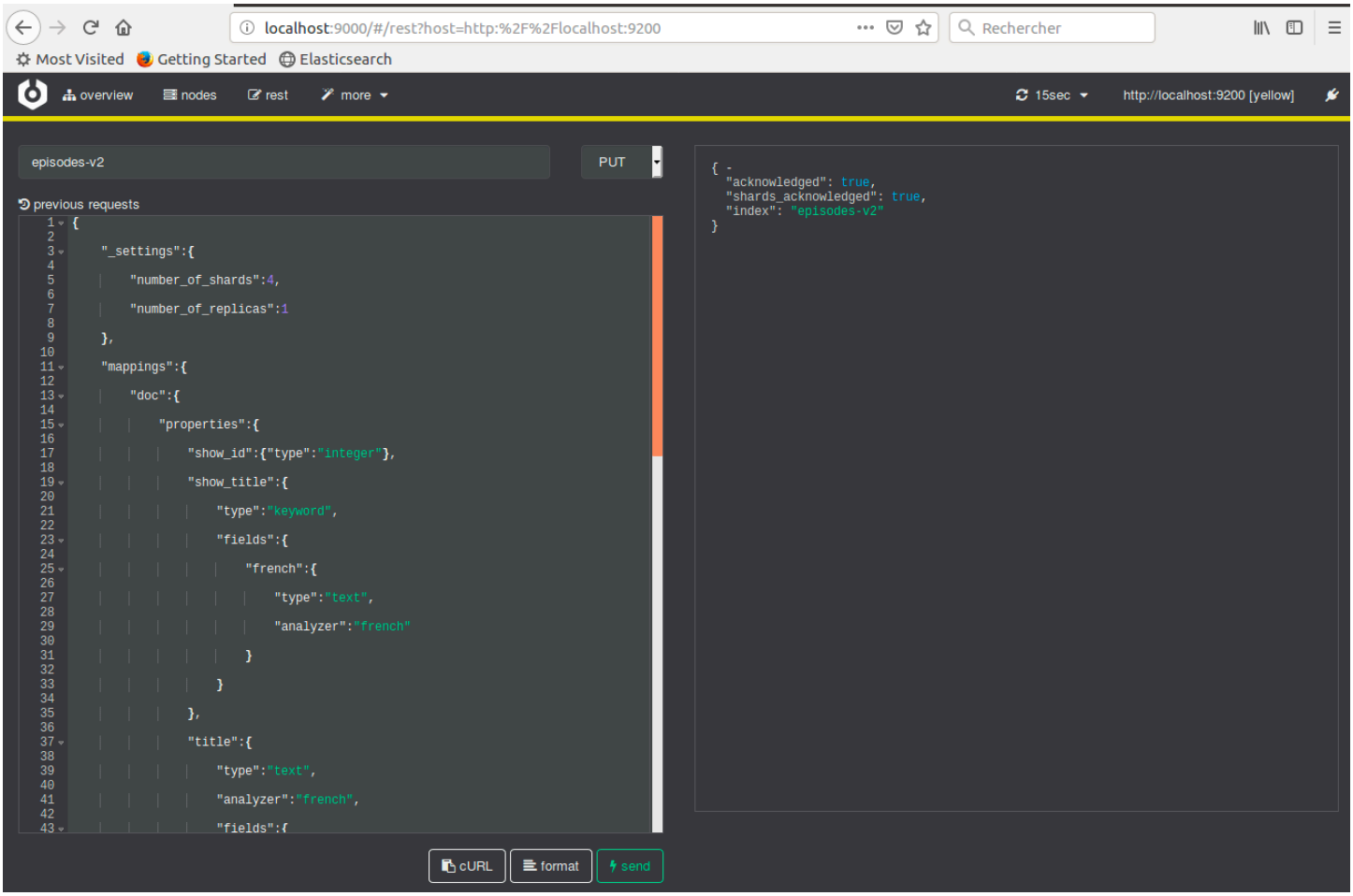
Nous allons donc réindexer nos deux index grâce à l'api Reindex. L'API Reindex s'occupe de transférer nos données vers un nouvel index existant, le plus souvent avec un mapping différent. Elasticsearch s'occupe dans ce cas-là de migrer le type des données. Tout est expliqué ici pour les mappings : Https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html

a) Compléter l'index shows (ci-dessous) en vous inspirant du mapping de l'index episodes-v2 (ci-dessous). Pour rappel, la création d'un index se fait avec un PUT <indexName> avec en payload le mapping de l'index.

Mapping episodes-v2

```
PUT episodes-v2

{
    "_settings":{
        "number_of_shards":4,
```

```
        "number_of_replicas":1
    },
    "mappings":{
        "doc":{
            "properties":{
                "show_id":{"type":"integer"},
                "show_title":{
                    "type":"keyword",
                    "fields":{
                        "french":{
                            "type":"text",
                            "analyzer":"french"
                        }
                    }
                },
                "title":{
                    "type":"text",
                    "analyzer":"french",
                    "fields":{
                        "english":{
                            "type":"text",
                            "analyzer":"english"
                        }
                    }
                },
                "description":{
                    "type":"text",
                    "analyzer":"french",
                    "fields":{
                        "english":{
                            "type":"text",
                            "analyzer":"english"
                        }
                    }
                },
                "season":{"type":"integer"},
                "episode":{"type":"integer"},
                "released_timestamp":{
                    "type":"date",
                    "format":"epoch_second"
                },
                "code":{
                    "type":"keyword"
                }
            }
        }
    }
}
```



Mapping shows-v2 (A COMPLETER et A CREER) :

```
{
        "_settings":{
           "number_of_shards":4,
           "number_of_replicas":1
        },
        "mappings":{
           "doc":{
              "properties":{
                 "show_id":{
                    "type":"integer"
                 },



                 "seasons":{
                    "type":"integer"
                 },



                 "genres":{
                    "type":"text",
                    "analyzer":"french",
                    "fields":{
                       "analyzed":{
                          "type":"text",
                          "fielddata": true,
                          "analyzer":"simple"
                       }
```

```
            }
        },
    }
  }
}
```

a) Quelle est la différence entre un champ de type Keyword et un champ de type Text ?

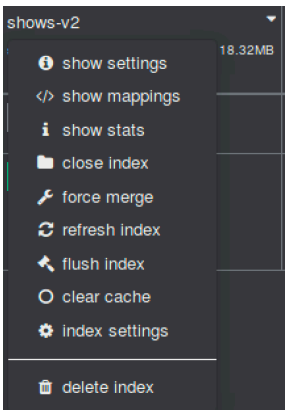
> ***Pour les champs analysés, il faut ajouter un sous-champ aux champs de type text avec un analyzer français et un sous champ analyzer anglais***
> ***Exemple dans la documentation avec un analyzer anglais :***
> ***Https://www.elastic.co/guide/en/elasticsearch/reference/5.2/multi-fields.html#_multi_fields_with_multiple_analyzers***

b) Grâce à l'endpoint de réindexation (Https://www.elastic.co/guide/en/elasticsearch/reference/6.2/docs-reindex.html), migrer les données vers nos nouveaux indexes.

> ***Il se peut qu'il reste un document dans l'ancien index qui ne corresponde pas au mapping que nous venons de faire. Dans ce cas-là, Elasticsearch renvoie une erreur et arrête la reindexation. Si vous avez cette erreur, garder l'ID du document (donné dans le retour) et supprimer ce document dans l'index d'origine. Il faudra par la suite supprimer l'index v2 (vous pouvez aussi le faire dans l'onglet Overview), le recréer et relancer la ré-indexation. C'est valable pour shows et episodes.***
>
> ***Vous pouvez utiliser un GET episodes/_search?q=episode:"episode" et GET shows/_search?q=title:"title" pour trouver les documents avant la reindexation et les supprimer avec un DELETE shows/doc/<ID>***

> ***COMMENCER PAR REINDEXER SHOWS ET NE PAS ATTENDRE LA REINDEXATION D'EPISODES CAR ELLE SERA TRES LONGUE (ne pas la relancer sinon DOUBLONS)***

Supprimer un index :



c) Retenter les requêtes de la partie II b). Le 3 premières requêtes Query DSL ne renvoient normalement plus de résultat. Essayer, par exemple la requête n°1, avec une recherche exacte sur le titre (« True Detective »). Compléter votre réponse de la question a.

d) Retenter la requête des séries star trek de la partie **II c).** Cette fois-ci la liste triée devrait s'afficher.

e) Tester l'analyzer français en regardant comment il interprète des recherches en langue française (Get _analyze). Exemple de phrase à analyser : « Comment réparer un vélo avec une brosse à dent ? »

> ***C'est cette analyse qui permet de calculer le score sur nos recherches. En le mettant en français, l'analyzer est ainsi plus à même de comprendre nos recherches et de retrouver des résultats concluants***

IV. AGREGATIONS

Nous allons maintenant faire quelques agrégations pour recueillir quelques statistiques sur nos données. Utiliser les index « v2 » que nous avons créés précédemment.

1) Faire ces différentes requêtes (mettre size = 0 dans la requête pour que ce soit plus rapide et lisible dans le résultat) :
   - Donner le nombre moyen de saisons par séries, ainsi que le nombre moyen de followers. Résultat : 720.89 de followers moyens et 2.31 saisons en moyenne
   - Donner le nombre moyen de followers pour les séries « Doctor who ». Résultat : 746.93
   - Donner le nombre moyen de followers par année de création des séries de 1968 à 2018. Exemple pour 1968 : 102.93
   - Donner les 3 networks ayant le plus de séries entre 1900 et 2018 et pour chacun, le nombre de séries créées par année. Résultat : Youtube en 2018 -> 4
   - Compter le nombre de séries par tranche de saisons (1-2, 2-3, 4-6, 7+). Indice : *range aggregation*. ***Résultat : entre 2 et 3 : 2386.***
   - Donner les 5 genres les plus populaires par année depuis 2017. Drama en 2017 : 496

V. SHARDING ET REPLICATIONS

Elasticsearch est un moteur de recherche portant une attention toute particulière sur la « résilience ». Nous allons tester et visualiser son comportement en cluster. Pour cela, nous allons lancer plusieurs instances d'Elasticsearch.

Tout d'abord, stopper ES et cerebro avec le script ***stop.sh*** et ajouter « ***node.max_local_storage_nodes: 4*** » à la fin du fichier *elasticsearch/config/elasticsearch.yml*. Relancer ensuite Cerebro et ES avec le script ***start.sh***

Dans le terminal, ouvrir le dossier elasticsearch et lancer deux fois la commande `bin/elasticsearch &` pour lancer de nouvelles instances. Penser à noter le PID des processus ainsi créés. Les nouveaux nœuds devraient apparaître dans Cerebro et l'initialisation des shards va commencer. Attendez que tout se stabilise avant de passer à la question b) !

a) A quoi correspondent les carrés verts et en pointillés sur Cerebro ? Pourquoi ne sont-ils pas attribués sur le même nœud ?
b) Que va-t-il se passer si on coupe notre instance master ? Va t'il y avoir une coupure d'accès à nos données ?
   Testez ensuite par vous-même :
   - Tuer le nœud master dont le pid est disponible dans le fichier *pids.pid* (kill -9 <ESPID>)
   - Vérifier que les requêtes fonctionnent toujours sur les ports 9201 et 9202
   - Connecter Cerebro sur HTTP://localhost:9202 et vérifier qu'un nouveau master a été élu

VI. KIBANA

Kibana est l'outil de visualisation (dataviz) des données d'Elasticsearch. C'est une façon simple d'explorer et d'afficher les données d'Elasticsearch. Il y a trois onglets principaux :

- Discover : Permet de fouiller dans la donnée en ajoutant des filtres (Query)
- Visualization : Faire des graphiques interactifs sur des agrégations
- Dashboard : assemblage de graphiques pour faire des tableaux de bord

Pour lancer Kibana, se rendre dans le dossier Kibana et exécuter la commande `bin/kibana &` . Il va alors se connecter au cluster ES sur le port 9200. Il faudra ensuite paramétrer un « index pattern » incluant l'index « shows-v2 » et un autre pour « episodes-v2 »

1) Essayer de reproduire les requêtes de la partie II sur l'onglet discover de Kibana
2) Essayer de faire des graphiques en reproduisant les requêtes de la partie IV

ANNEXES

STRUCTURE DES DOCUMENTS « SHOWS »

Champ	Type	Analysé ?
show_id	**Numeric**	
title	**Text**	**oui**
description	**Text**	**oui**
seasons	**Numeric**	
episodes	**Numeric**	
follow_count	**Numeric**	
creation_year	**Date (year)**	
genres	**Text**	**Oui <- simple analyzer**
network	**Keyword**	**non**

Structure des documents « Épisodes »

Champ	Type	Analysé ?
episode_id	**Numeric**	
show_id	**Numeric**	
show_Title	**Keyword**	**Oui (en sous-champ)**
code	**Text**	**non**
season	**Numeric**	
episode	**Numeric**	
title	**Text**	**oui**
description	**Text**	**oui**
released_timestamp	**Date (epoch_second)**	