Elasticsearch

### **1. Principes fondamentaux : Qu'est-ce qu'Elasticsearch et quels sont ses principaux cas d'utilisation ?**

Elasticsearch est un moteur de recherche et d'analyse distribué, open-source, basé sur Lucene. Il est conçu pour l’indexation et la recherche rapide de grandes quantités de données en quasi-temps réel.

Principaux cas d'utilisation :

* Recherche de texte plein : Moteur de recherche pour sites web, applications et bases de données.
* Analyse de logs : Centralisation et analyse des logs (ex. ELK Stack : Elasticsearch, Logstash, Kibana).
* Monitoring et observabilité : Suivi des performances d’applications et de serveurs.
* Analyse de données et BI : Exploration et visualisation de données structurées et non structurées.
* Systèmes de recommandation : Personnalisation basée sur l’analyse de texte et de comportements.

### **2. Architecture et fonctionnement : Comment fonctionne l’architecture d’Elasticsearch ?**

Elasticsearch est basé sur une architecture distribuée :

* Cluster : Ensemble de nœuds travaillant ensemble.
* Nœud : Unité de calcul qui stocke les données et exécute les requêtes.
* Index : Collection de documents partageant une structure commune.
* Shards : Un index est divisé en plusieurs shards pour permettre la répartition des données et la parallélisation des requêtes.
* Réplication : Chaque shard a des copies de secours pour la tolérance aux pannes.

Mode de stockage des données :

Elasticsearch stocke les documents sous format JSON. Chaque document est indexé via Lucene, qui crée une structure optimisée pour la recherche rapide.

### **3. Langage et requêtage : Comment s’effectuent les recherches et les requêtes dans Elasticsearch ?**

Elasticsearch utilise Query DSL, un langage basé sur JSON, pour construire des requêtes complexes.

Types de requêtes :

* Recherches full-text : Analyse et scoring des résultats via des tokens et des analyzers.
* Filtres booléens : Combinaison de critères avec must, should, must\_not, filter.
* Aggregation Framework : Regroupement et statistiques avancées.
* Scoring et pertinence : Basé sur TF-IDF et BM25.

*Exemple de requête :*

{

"query": {

"bool": {

"must": [

{ "match": { "title": "Elasticsearch" } }

],

"filter": [

{ "range": { "date": { "gte": "2024-01-01" } } }

]

}

}

}

### **4. Performance et scalabilité : Quels sont les mécanismes d’optimisation des performances ?**

Sharding : Division d’un index en plusieurs shards pour la répartition de la charge.

Réplication : Copies redondantes des shards pour la haute disponibilité.

Caching : Elasticsearch met en cache certaines requêtes pour accélérer les recherches.

Refresh Interval : Ajustement de la fréquence d’indexation en fonction du besoin en temps réel.

Compression des données : Stockage efficace des index pour réduire l’empreinte mémoire.

Tuning des mappings : Adaptation des types de champs pour optimiser la recherche.

### **5. Sécurité et administration : Quelles sont les bonnes pratiques pour sécuriser et administrer un cluster Elasticsearch ?**

***Sécurisation des accès :***

Activation de TLS/SSL pour chiffrer les communications.

Utilisation de X-Pack Security pour gérer l’authentification et les rôles.

Limitation des accès avec des firewalls et des IP whitelists.

***Protection des données :***

Gestion des snapshots pour les sauvegardes.

Paramétrage des rétentions d’index avec Index Lifecycle Management (ILM).

***Monitoring et maintenance :***

Utilisation de Kibana, Elastic Stack Monitoring pour suivre les performances.

Répartition des rôles des nœuds : master, data, ingest pour une meilleure stabilité.

***Optimisation des performances :***

Éviter les mappings dynamiques non contrôlés.

Supprimer ou optimiser les index non utilisés.

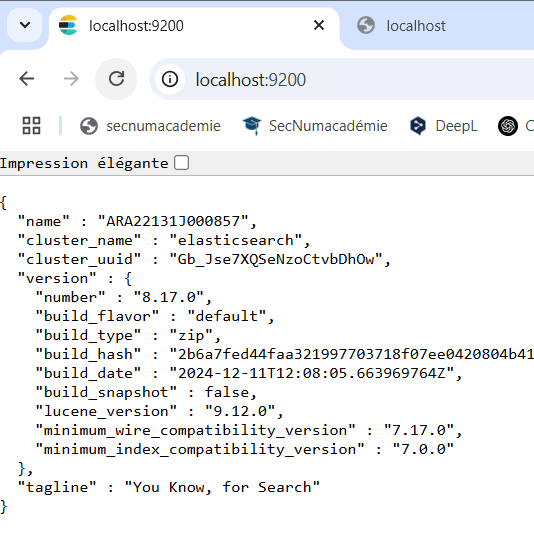
Ajuster les paramètres de heap size et de garbage collection pour éviter les blocages mémoire.

## 

## TP1 : Installation ElasticSearch

1. **Installation d’Elasticsearch et Kibana :**

* téléchargement du dossier zip pour Windows
* lancement du fichier “elasticsearch.bat” via la cmd
* ouverture du navigateur : localhost:9200



1. **Importation des données :**

Service bulk avec Curl :

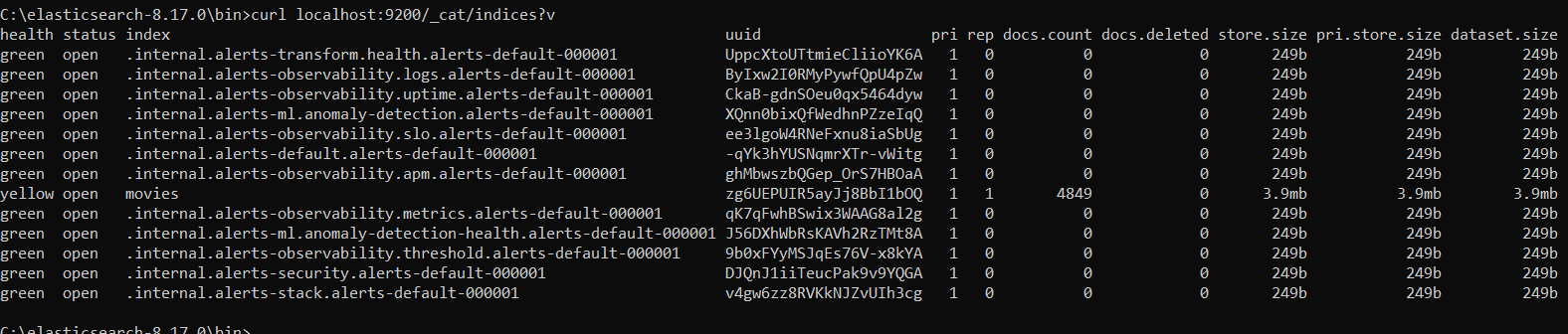
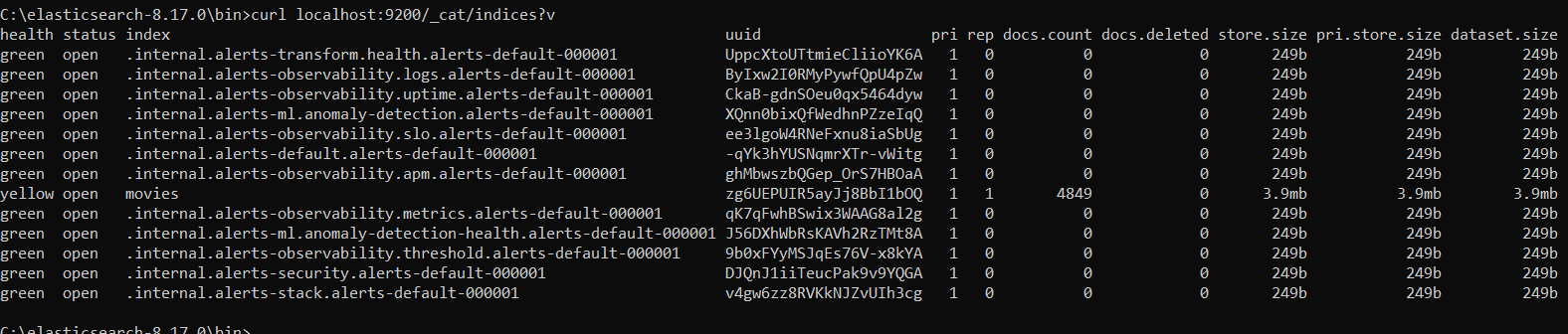
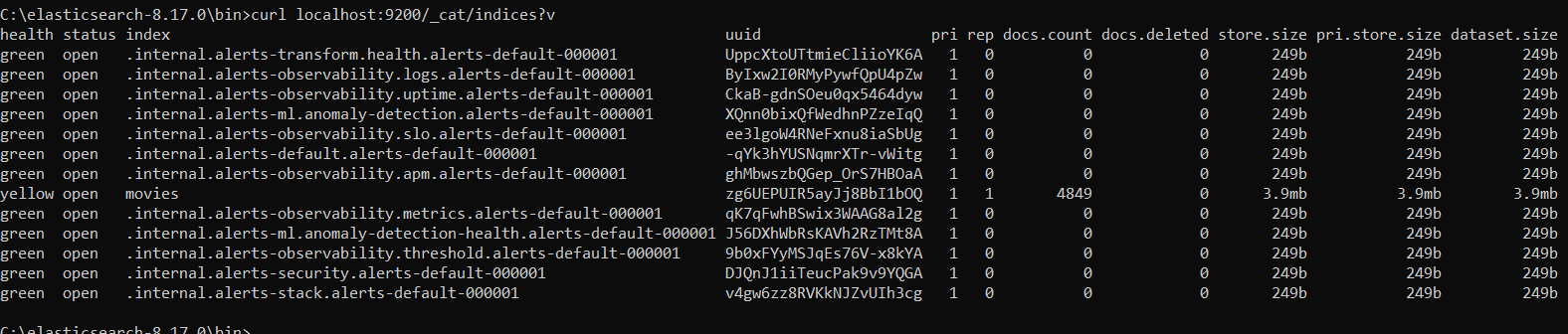
***curl -XPUT localhost:9200/bulk -H "Content-Type: application/json" --data-binary @movies\_elastic.json***

Attention j’ai dû modifier le fichier .json car le paramètre “\_type” n’est plus supporté par la version de ElasticSearch installée.

## 

## 

* vérifier la liste des indices : curl localhost:9200/\_cat/indices?v



## 

## TP2 : Intro Queries

Requêtes simples avec la méthode HTTP GET et le paramètre q=

* Recherche de mots dans l’ensemble de chaque document :

***http://localhost:9200/movies/\_search?q=alien***

***OU***

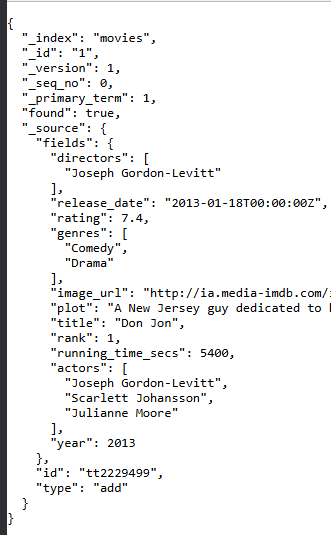
***curl -X GET "http://localhost:9200/movies/\_search?q=alien***

****

* Pour vérifier les données et afficher par exemple le film avec l’id:1 il faut exécuter la commande suivante :

***curl -X GET*** [***http://localhost:9200/movies/\_doc/1***](http://localhost:9200/movies/_doc/1)

(depuis les versions Elasticsearch 7.X et 8.X, il faut remplacer /index/\_type/id par index/\_doc/id)



* Recherche de mots dans le titre de chaque document :

***http://localhost:9200/movies/\_search?q=fields.title:alien***

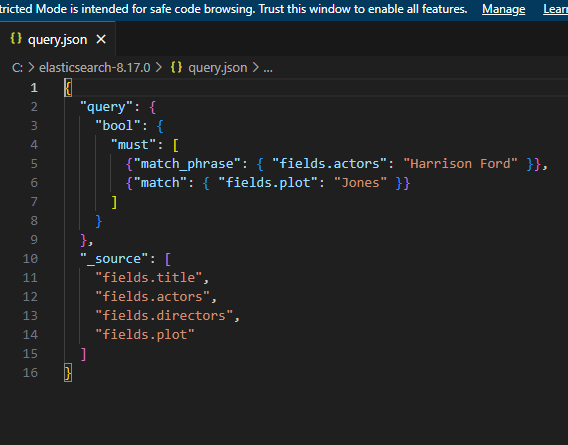
***OU***

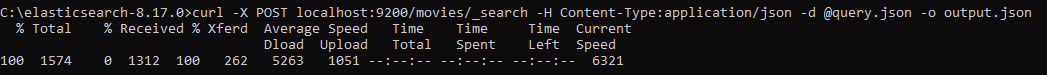
***curl -X GET "http://localhost:9200/movies/\_search?q=fields.title:alien"***

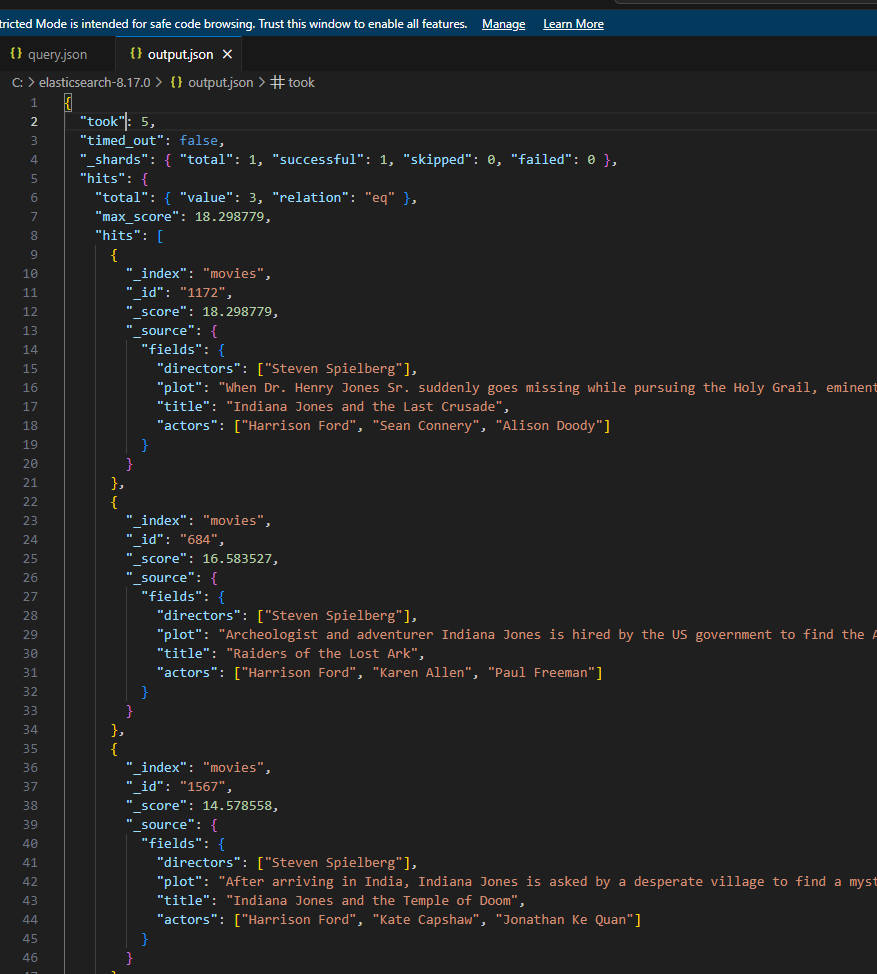
d’après le mapping, pour récupérer des données qui figure dans le titre il faut écrire fields.title

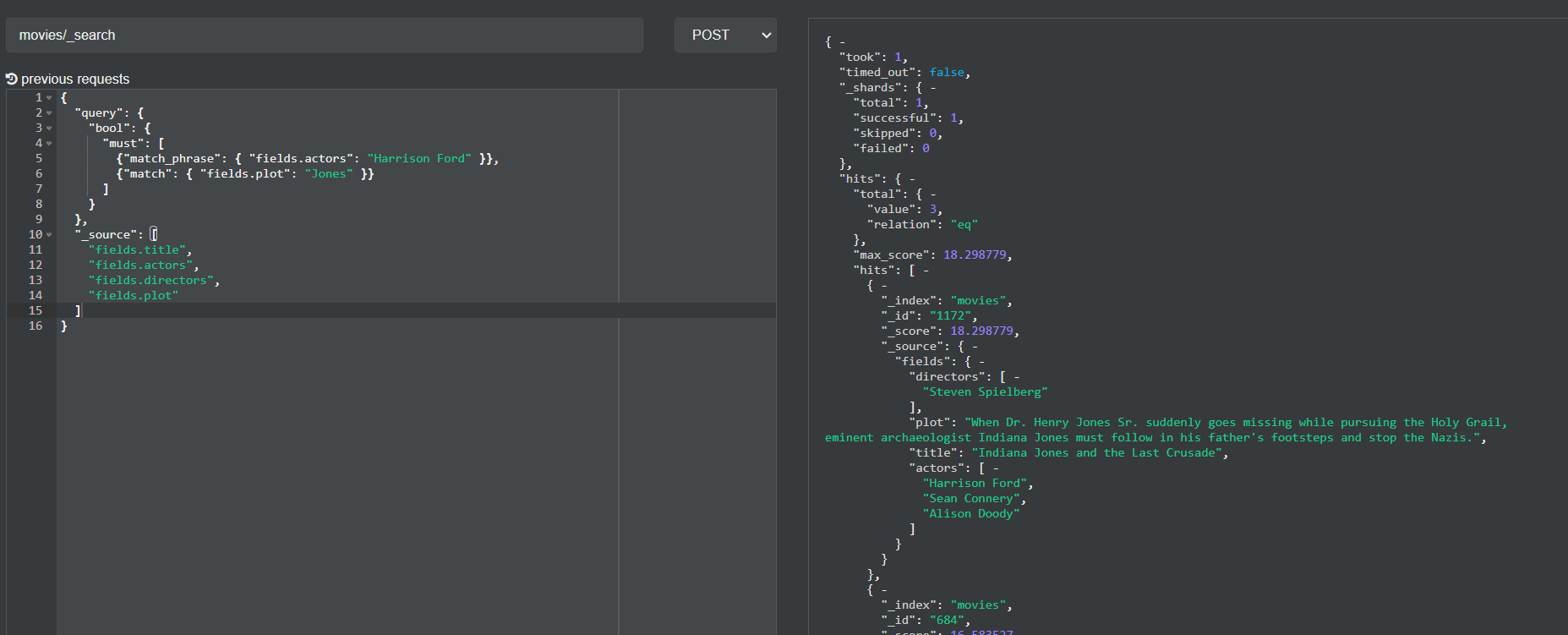
Requête simple avec la méthode HTTP POST :

* création d’une requête que l’on nomme query.json



* exécution de la commande dans le terminal : 
* récupération de la réponse à la requête sous format json (création automatique du fichier output.json)

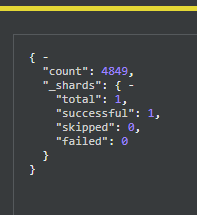


Méthode POST via Cerebro :

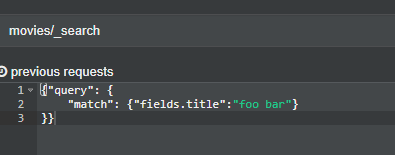
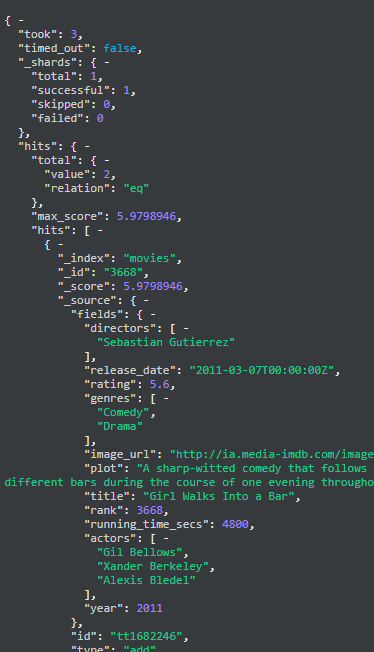
Requêtes simples en utilisant des outils de dev comme Cerebro ou Kibana :

* Compter le nombre de documents :

***http://localhost:9200/movies/\_count***

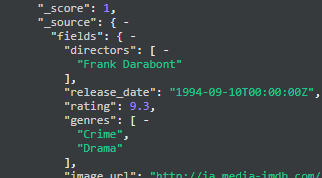
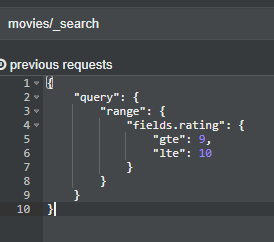


* Requête de matching via Cerebro :

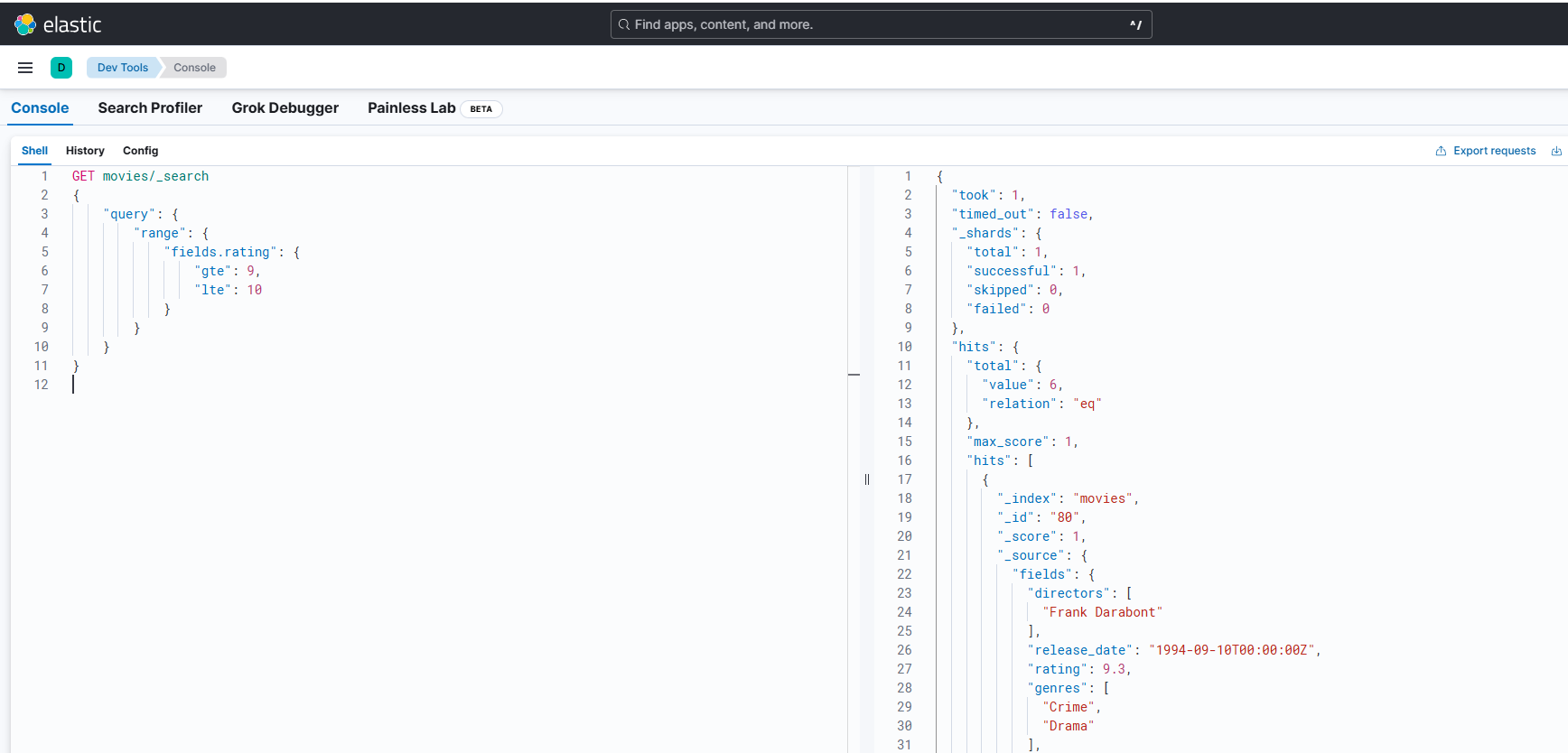


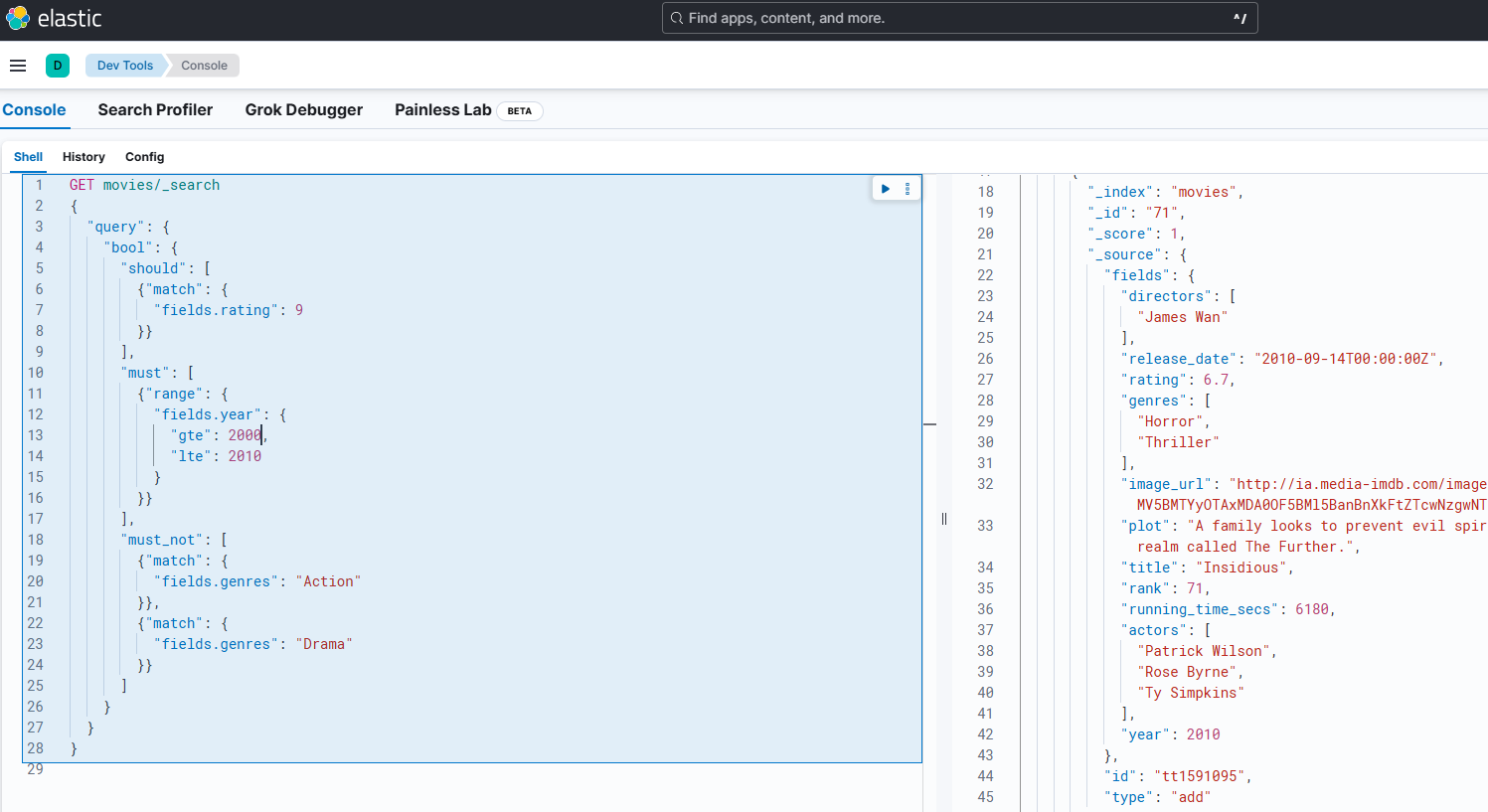
GET

* Requêtes d’intervalles (range) :



Test via Kibana :



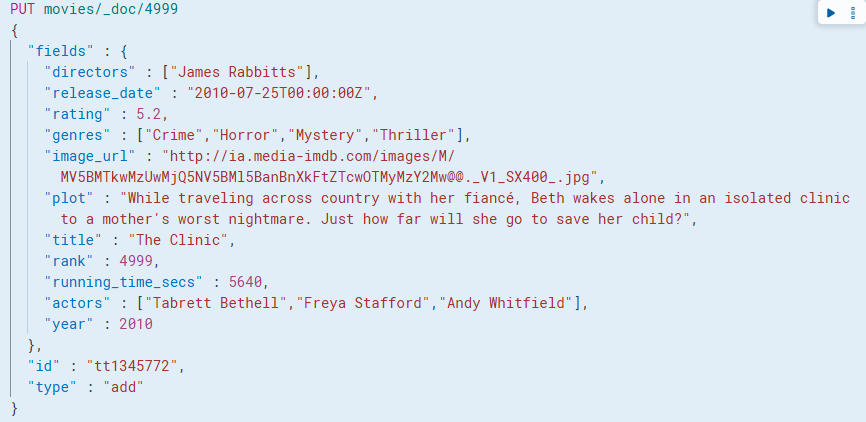
* Requêtes simples : must & should
* Requêtes simples : filter



## 

## TP3 : Gestion d’index + CRUD

PUT : cette commande remplace le document précédent et donc écrase les anciennes données. Elle permet également de créer un nouveau document





POST : permet de modifier la valeur d’un champ

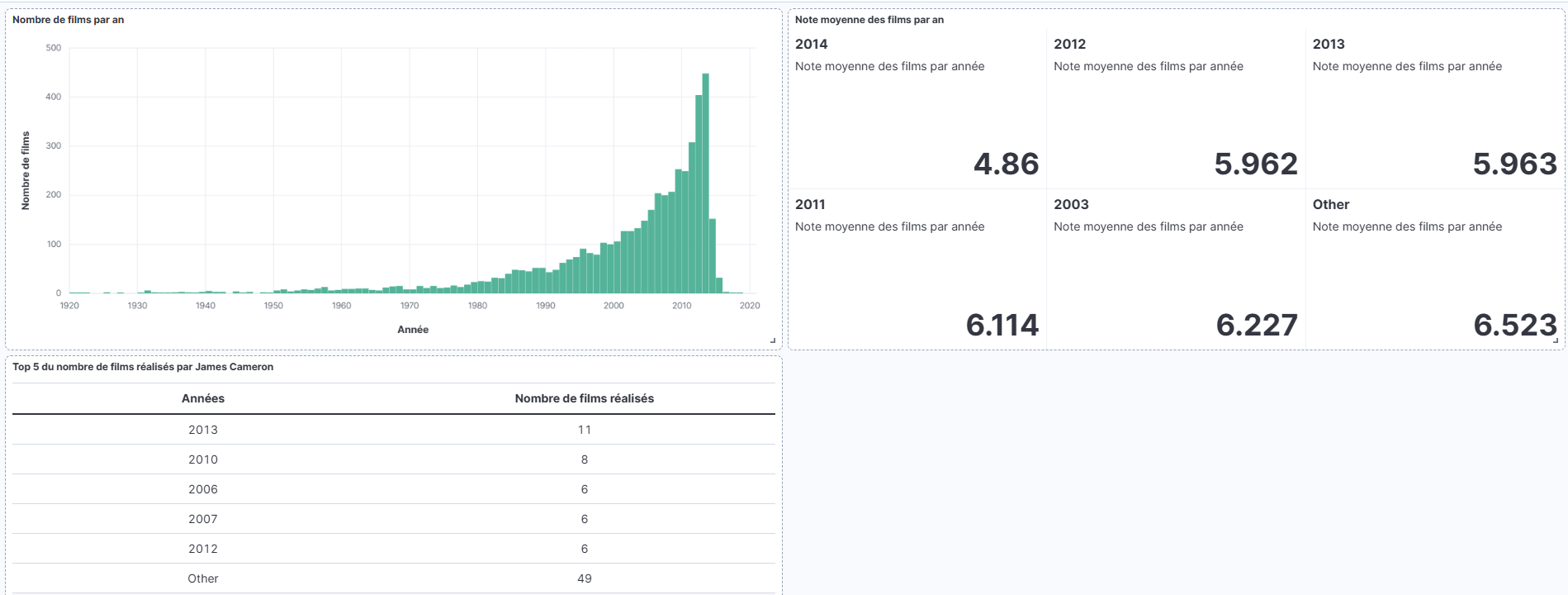


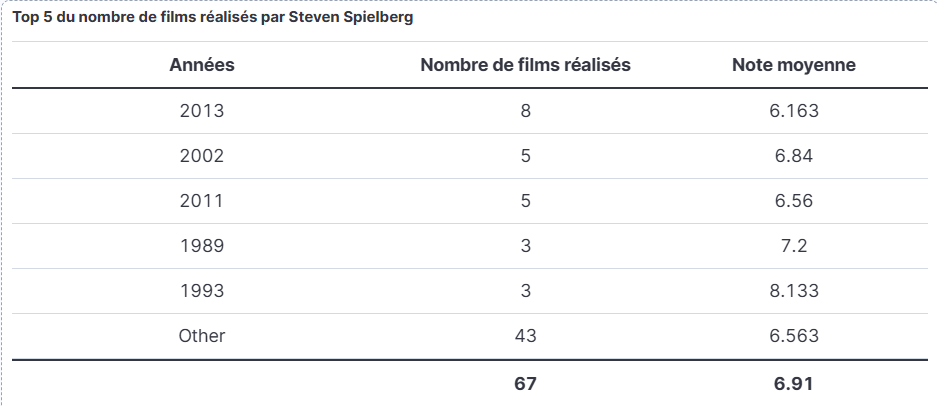
DELETE : supprime un document

DELETE movies/\_doc/4999



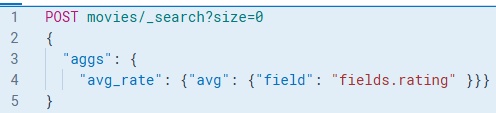
Création de graphes avec Kibana:





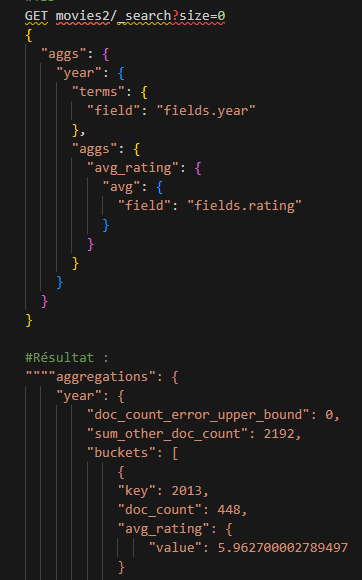
## TP4 : Exploitation des données

1. Agrégations de métriques





1. Agrégations de compartiments



1. Visualisations via Kibana

