Travaux Pratiques sur Elasticsearch

TP sur Elasticsearch : De la Manipulation de Données au Mapping Avancé

Objectif : Comprendre et pratiquer les opérations de base et avancées sur Elasticsearch, incluant l'importation de données, la recherche, l'agrégation, et la personnalisation du mapping.

Prérequis :

- Elasticsearch installé et actif sur localhost.
- Fichiers `movies_elastic.json` et `mapping_movies` disponibles localement.
- `curl` installé pour interagir avec l'API Elasticsearch ou `elasticvue`

Basic Authentication

If your Elasticsearch server uses Basic Authentication, you would include the username and password in the `curl` command. Here's how you can modify your command to include Basic Authentication credentials:

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

Exercice 1 : Importation des données dans un Index Existant

- **Tâche :** Importez les données du fichier `movies_elastic.json` dans l'index `movies`.
- **Commande :**

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

- **Explication :**
  Cette commande envoie une requête PUT à l'API `_bulk` d'Elasticsearch pour importer plusieurs documents. L'option `--data-binary` garantit que les données sont transmises exactement comme elles sont sans modification.

Exercice 2 : Création d'un Nouvel Index avec Mapping Personnalisé

- **Tâche :** Créez un index `movies2` avec un mapping spécifique pour traiter certains champs comme des données brutes (raw).

- **Commande :**

```
curl -XPUT -H "Content-Type: application/json" localhost:9200/movies2 -d
@mapping_movies/mapping.json
```

- **Explication :** Configure le mapping de `movies2` pour que les champs comme `actors`, `directors`, et `genres` soient indexés comme des données brutes, facilitant des requêtes précises sans décomposition en mots-clés.

Exercice 3 : Importation des données avec le Mapping Personnalisé

- **Tâche :** Importez les données modifiées dans le nouvel index `movies2` en utilisant le mapping personnalisé.
- **Commande :**

```
curl -XPUT -H "Content-Type: application/json" localhost:9200/_bulk
--data-binary @mapping_movies/movies_elastic2.json
```

- **Explication :** Importe en masse les données dans `movies2` en utilisant l'API `_bulk`, assurant que les données sont indexées selon le nouveau mapping défini.

Exercice 4 : Vérification et Recherche Avancée

- **Tâche :** Vérifiez le mapping et exécutez des requêtes avancées pour exploiter le nouveau schéma (Quels sont les films réalisés par Ron Howard).
- **Commandes :**

```
curl -XGET "http://localhost:9200/movies2/_search?pretty" -H 'Content-Type:
application/json' -d'
{
  "query": {
    "bool": {
      "must": [
```

```
        { "match": { "directors.raw": "Ron Howard" }}
      ]
    }
  }
}'
```

- **Explication :**

  La première commande permet de vérifier le mapping appliqué à `movies2`. La deuxième commande utilise ce mapping pour rechercher spécifiquement des films par "Ron Howard", démontrant l'utilité d'un traitement `raw` des noms.

Exercice 5 : Aggrégation des Données

- **Tâche :** Effectuez une agrégation pour compter le nombre de films par genre dans l'index `movies`.
- **Commande :**

```
curl -XGET "http://localhost:9200/movies/_search?pretty" -H 'Content-Type:
application/json' -d'
{
  "size": 0,
  "aggs": {
    "genres": {
      "terms": { "field": "genres.keyword" }
    }
  }
}'
```

- **Explication :**

  Cette commande effectue une agrégation pour compter les occurrences de chaque genre, illustrant comment analyser des données agrégées pour obtenir des insights.

Exercice 6 : Recherche Simple (Requête URI) :

- **Tâche :** Recherchez tous les films contenant "Star Wars" dans leur titre.
- **Commande :**

```
curl -XGET 'http://localhost:9200/movies2/_search?q=Star+Wars&pretty'
```

- **Explication :**

  Cette URL lance une recherche simple sur l'index `movies2` pour tous les films dont le titre contient "Star Wars". La requête est simple et directe, utilisant le paramètre de requête `?q=`.

Exercice 7 : Filtrage des Résultats (Requête URI) :

- **Tâche :** Affinez la recherche pour inclure uniquement les films dont George Lucas est le réalisateur.
- **Commande :**

```
curl -XGET 'http://localhost:9200/movies2/_search?q=fields.title:Star Wars AND fields.directors:George Lucas&pretty'
```

- **Explication :**

  Cette commande filtre les résultats pour montrer seulement ceux où "George Lucas" est listé comme réalisateur. La requête utilise l'opérateur logique `AND` pour s'assurer que les deux conditions sont remplies.

Exercice 8 : Requête Booléenne Complexes(Requête URI) :

- **Tâche :** Trouvez les films où Harrison Ford a joué et dont le plot mentionne "Jones", mais sans mentionner "Nazis".
- **Commande :**

```
curl -XGET 'http://localhost:9200/movies2/_search?q=fields.actors:Harrison Ford AND fields.plot:Jones AND -fields.plot:Nazis&pretty'
```

- **Explication :**

  Cette requête booléenne combine des conditions positives et une négation pour filtrer les films selon des critères spécifiques. Cela montre comment

combiner plusieurs champs de recherche et utiliser la négation pour exclure certains résultats.

Exercice 9 : Pagination des Résultats(Requête URI) :

- **Tâche :** Recherchez les films avec Harrison Ford, en limitant les résultats aux 20 premiers.
- **Commande :**

```
curl -XGET 'http://localhost:9200/movies2/_search?q=fields.actors:Harrison Ford&size=20&pretty'
```

- **Explication :**

  Cette URL ajoute un paramètre de pagination `size` pour contrôler le nombre de résultats retournés. Cela est utile pour gérer de grands volumes de données et pour explorer les résultats par pages.

Exercice 10 : Requête Simple Match (Requête body JSON) :

- **Tâche :** Recherchez tous les films dont le titre contient "Star Wars".
- **Commande :**

  JSON File (query1.json):

```
{
  "query": {
    "match": {
      "fields.title": "Star Wars"
    }
  }
}
```

- cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'http://localhost:9200/movies2/_search?pretty' -d @query1.json
```

- **Explication** :

  Cette commande crée un fichier JSON contenant la requête de recherche, puis utilise `curl` pour envoyer cette requête à Elasticsearch. Elle montre comment effectuer une recherche simple par correspondance de texte.

Exercice 11 : Requête Booléenne avec Should et Must (Requête body JSON):

- **Tâche :** Trouvez les films où George Lucas est mentionné comme réalisateur et le titre contient "Star Wars".
- **Commande :**

  JSON File (query2.json):

```
{
  "query": {
    "bool": {
      "should": [
        {
          "match": {
            "fields.title": "Star Wars"
          }
        },
        {
          "match": {
            "fields.directors": "George Lucas"
          }
        }
      ]
    }
  }
}
```

  cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'http://localhost:9200/movies2/_search?pretty' -d @query2.json
```

- **Explication :**

  Cette requête booléenne combine les conditions `should` pour une correspondance souple et `must` pour une correspondance obligatoire, illustrant comment mélanger des critères de recherche flexibles et stricts.

Exercice 12 : Match Phrase et Filtres Complexes (Requête body JSON):

- **Tâche :** Recherchez les films qui mentionnent précisément "Star Wars" suivi par "George Lucas" comme réalisateur.
- **Commande :**

  JSON File (query3.json):

```
{
  "query": {
    "bool": {
      "should": [
        {
          "match_phrase": {
            "fields.title": "Star Wars"
          }
        },
        {
          "match": {
            "fields.directors": "George Lucas"
          }
        }
      ]
    }
  }
}
```

  cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'http://localhost:9200/movies2/_search?pretty' -d @query3.json
```

- **Explication :**

  Utilisation de `match_phrase` pour garantir que les mots apparaissent dans l'ordre exact dans le champ. Ceci est utile pour des recherches de phrases spécifiques plutôt que de simples mots-clés.

Exercice 13 : Exclusion de Termes avec Must Not (Requête body JSON):

- **Tâche :** Trouvez des films où Harrison Ford joue, qui mentionnent "Jones" mais excluent "Nazis".
- **Commande :**

  JSON File (query.json):

```
{
  "query": {
    "bool": {
      "should": [
        {
          "match": {
            "fields.actors": "Harrison Ford"
          }
        },
        {
          "match": {
            "fields.plot": "Jones"
          }
        }
      ],
      "must_not": {
        "match": {
          "fields.plot": "Nazis"
        }
      }
    }
  }
}
```

  cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'http://localhost:9200/movies2/_search?pretty' -d @query.json
```

- **Explication :**

  Cette requête combine `should` pour les conditions souples et `must_not` pour exclure les documents contenant certains mots, permettant des recherches précises et ciblées.

Exercice 14 : Filtrage Simple avec "Range" (Requête body JSON):

- **Tâche :** Trouvez les films de James Cameron avec un rang inférieur à 1000.
- **Commande :**

  JSON File (query.json):

```
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "fields.directors": "James Cameron"
          }
        },
        {
          "range": {
            "fields.rank": {
              "lt": 1000
            }
          }
        }
      ]
    }
  }
}
```

  cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'http://localhost:9200/movies2/_search?pretty' -d @query.json
```

- **Explication :**

  Cette requête combine un filtre de correspondance exacte pour les films

dirigés par James Cameron avec un filtre de plage pour sélectionner les films dont le rang est inférieur à 1000.

Exercice 15 : Filtrage Combiné avec Match Phrase et Range (Requête body JSON):

- **Tâche :** Recherchez précisément les films de James Cameron dont le rang est inférieur à 1000.
- **Commande :**

  JSON File (query.json):

```
{
  "query": {
    "bool": {
      "must": [
        {
          "match_phrase": {
            "fields.directors": "James Cameron"
          }
        },
        {
          "range": {
            "fields.rank": {
              "lt": 1000
            }
          }
        }
      ]
    }
  }
}
```

cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'http://localhost:9200/movies2/_search?pretty' -d @query.json
```

- **Explication :**

  L'utilisation de `match_phrase` permet de s'assurer que le terme exact "James Cameron" est recherché dans les données, en combinaison avec un filtre de plage sur le rang.

Exercice 14 : Filtrage Simple avec "Range"

- *Type de Correspondance : match*
- *Utilise le critère match, qui effectue une recherche sur le texte pour trouver des correspondances approximatives. Cela signifie qu'il peut trouver des résultats où "James Cameron" apparaît dans le champ spécifié, mais aussi potentiellement d'autres mots proches ou des variations selon le tokenizer utilisé par Elasticsearch.*

Exercice 15 : Filtrage Combiné avec Match Phrase et Range

- *Type de Correspondance : match_phrase*
- *Utilise le critère match_phrase, qui requiert une correspondance exacte de la phrase "James Cameron". Cette méthode garantit que le champ contient précisément les termes "James Cameron" dans l'ordre spécifié, sans mots supplémentaires entre eux. Cela est utile lorsque la précision de l'expression exacte est cruciale.*

Résumé des différences :

- *Approche de Recherche : match vs. match_phrase*
  - *match est plus flexible et peut retourner des résultats avec des correspondances partielles ou proches.*
  - *match_phrase est strict et ne retournera des résultats que lorsque l'expression exacte apparaît dans le champ.*

Les deux méthodes ont leurs utilisations selon le besoin de précision dans la recherche. Pour des recherches où l'exactitude de l'expression est essentielle (par exemple, des noms propres ou des termes

techniques), match_phrase est préférable. Pour des recherches plus générales où la flexibilité est avantageuse, match pourrait être suffisant.

Exercice 16 : Requête avec Exclusion de Genres

- **Tâche :** Trouvez les films de James Cameron avec une note supérieure à 5, mais qui ne sont ni d'action ni dramatiques.
- **Commande :**

  JSON File (query.json):

```
{
  "query": {
    "bool": {
      "should": [
        {
          "match": {
            "fields.directors": "James Cameron"
          }
        }
      ],
      "must": [
        {
          "range": {
            "fields.rating": {
              "gte": 5
            }
          }
        }
      ],
      "must_not": [
        {
          "match": {
            "fields.genres": "Action"
          }
        },
        {
          "match": {
            "fields.genres": "Drama"
          }
        }
      ]
```

```
    }
  }
}
```

cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'http://localhost:9200/movies2/_search?pretty' -d @query.json
```

- **Explication :**

  Cette requête montre comment utiliser des filtres "must_not" pour exclure certains genres tout en recherchant des films avec une note élevée, mettant en avant la flexibilité des requêtes booléennes.

Exercice 17 : Recherche Temporelle avec Range sur Dates

- **Tâche :** Recherchez les films de J.J. Abrams sortis entre 2010 et 2015.
- **Commande :**

  JSON File (query.json):

```
{
  "query": {
    "bool": {
      "must": {
        "match": {
          "fields.directors": "J.J. Abrams"
        }
      },
      "filter": {
        "range": {
          "fields.release_date": {
            "from": "2010-01-01",
            "to": "2015-12-31"
          }
        }
      }
    }
  }
}
```

cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'http://localhost:9200/movies2/_search?pretty' -d @query.json
```

- **Explication :**

  Cette requête filtre les films en fonction de leur date de sortie, utilisant un filtre "range" pour définir une période spécifique, démontrant la puissance des requêtes temporelles dans Elasticsearch.

Exercice 18 : Agrégation Simple - Nombre de Films par Année

- **Tâche :** Calculez le nombre de films sortis chaque année.
- **Commande :**

  JSON File (query.json):

```
{
  "size": 0,
  "aggs": {
    "nb_par_annee": {
      "terms": {
        "field": "fields.year"
      }
    }
  }
}
```

  cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'http://localhost:9200/movies2/_search?pretty' -d @query.json
```

- **Explication :**

  Cette requête crée une agrégation qui compte le nombre de films par année, utilisant la clé "terms" pour grouper par année. La réponse inclut un compteur pour chaque année.

Exercice 19 : Calcul de la Note Moyenne des Films

- **Tâche :** Déterminez la note moyenne des films dans l'ensemble de la base de données.
- **Commande :**

  JSON File (query.json):

```
{
  "size": 0,
  "aggs": {
    "note_moyenne": {
      "avg": {
        "field": "fields.rating"
      }
    }
  }
}
```

  cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'localhost:9200/movies2/_search?pretty' -d @query.json
```

- **Explication :**

  Cette commande utilise la fonction d'agrégation "avg" pour calculer la note moyenne des films, fournissant une statistique de base pour l'évaluation globale des films.

Exercice 20 : Agrégation Combinée sur les Films de George Lucas

- **Tâche :** Affichez la note moyenne et le rang moyen des films de George Lucas.
- **Commande :**

  JSON File (query.json):

```
{
  "query": {
    "match_phrase": {
```

```
      "fields.directors": "George Lucas"
    }
  },
  "size": 0,
  "aggs": {
    "note_moyenne": {
      "avg": {
        "field": "fields.rating"
      }
    },
    "rang_moyen": {
      "avg": {
        "field": "fields.rank"
      }
    }
  }
}
```

cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'localhost:9200/movies2/_search?pretty' -d @query.json
```

- **Explication :**

  Cette requête filtre les films par le réalisateur George Lucas, puis calcule la note moyenne et le rang moyen, utilisant une combinaison de fonctions d'agrégation.

Exercice 21 : Agrégation de la Note Moyenne des Films par Année

- **Tâche :** Calculez la note moyenne des films par année et triez les résultats par note moyenne descendante.
- **Commande :**

  JSON File (query.json):

```
{
  "size": 0,
```

```
  "aggs": {
    "group_year": {
      "terms": {
        "field": "fields.year",
        "order": {
          "note_moyenne": "desc"
        }
      },
      "aggs": {
        "note_moyenne": {
          "avg": {
            "field": "fields.rating"
          }
        },
        "note_min": {
          "min": {
            "field": "fields.rating"
          }
        },
        "note_max": {
          "max": {
            "field": "fields.rating"
          }
        }
      }
    }
  }
}
```

cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'localhost:9200/movies2/_search?pretty' -d @query.json
```

- **Explication :**

  Cette commande agrège les données par année, puis calcule la note moyenne, minimale et maximale pour chaque année. Elle tri également les années par la note moyenne en ordre décroissant pour identifier les années avec les meilleurs films en moyenne.

Exercice 22 : Agrégation par Plages de Notes

- **Tâche :** Calculez le nombre de films par plage de notes de taille 2 (0-1.9, 2-3.9, etc.).
- **Commande :**

JSON File (query.json):

```
{
  "size": 0,
  "aggs": {
    "group_year": {
      "terms": {
        "field": "fields.year",
        "order": {
          "note_moyenne": "desc"
        }
      },
      "aggs": {
        "note_moyenne": {
          "avg": {
            "field": "fields.rating"
          }
        },
        "note_min": {
          "min": {
            "field": "fields.rating"
          }
        },
        "note_max": {
          "max": {
            "field": "fields.rating"
          }
        }
      }
    }
  }
}
```

cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'localhost:9200/movies2/_search?pretty' -d @query.json
```

- **Explication :**

Cette commande crée une agrégation de type "range" pour compter le nombre de films qui tombent dans des plages spécifiques de notes. Elle aide à analyser la distribution des évaluations des films.

Exercice 23 : Agrégation sur les Genres de Films

- **Tâche :** Comptez le nombre de films par genre, en utilisant le champ "raw" pour un regroupement précis.

Commande :

JSON File (agg_genres.json):

```
{
  "size": 0,
  "aggs": {
    "nb_per_genres": {
      "terms": {
        "field": "fields.genres.raw"
      }
    }
  }
}
```

cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'http://localhost:9200/movies2/_search?pretty' -d @agg_genres.json
```

- **Explication :**

  Cet exercice montre comment agréger sur des listes de valeurs en utilisant des champs non analysés pour éviter la tokenisation et obtenir un comptage précis des genres.

Exercice 24 : Agrégation sur les Acteurs avec Statistiques

- **Tâche :** Calculez la note moyenne, ainsi que le rang minimum et maximum pour les films de chaque acteur.
- **Commande :**

  JSON File (agg_actors.json):

```
{
  "size": 0,
  "aggs": {
    "group_actors": {
      "terms": {
        "field": "fields.actors.raw"
      },
      "aggs": {
        "note_moyenne": {
          "avg": {
            "field": "fields.rating"
          }
        },
        "rang_min": {
          "min": {
            "field": "fields.rank"
          }
        },
        "rang_max": {
          "max": {
            "field": "fields.rank"
          }
        }
      }
    }
  }
}
```

cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'http://localhost:9200/movies2/_search?pretty' -d @agg_actors.json
```

- **Explication :**

Cet exercice combine plusieurs fonctions d'agrégation sous un même acteur pour obtenir des statistiques détaillées sur leur filmographie. Il illustre l'efficacité de l'agrégation nested pour les données complexes.

Exercice 25 : Tri des Résultats d'Agrégation

- **Tâche :** Triez les acteurs par leur note moyenne décroissante.
- **Commande :**

  JSON File (agg_actors_sort.json):

```
{
  "size": 0,
  "aggs": {
    "group_actors": {
      "terms": {
        "field": "fields.actors.raw",
        "order": {
          "note_moyenne": "desc"
        }
      },
      "aggs": {
        "note_moyenne": {
          "avg": {
            "field": "fields.rating"
          }
        }
      }
    }
  }
}
```

cURL Command:

```
curl -XGET -H "Content-Type: application/json"
'localhost:9200/movies2/_search?pretty' -d @agg_actors_sort.json
```

- **Explication :**

  Cette commande ajoute un critère de tri à l'agrégation pour ordonner les acteurs selon la note moyenne de leurs films, permettant une analyse plus raffinée de leur performance.

Exercice 26 : Compréhension du Sharding et de la Réplication

- **Tâche :** Analyser la répartition des shards et des replicas dans un index.
- **Commande :**

```
# Créez un index avec une configuration spécifique de shards et replicas
curl -X PUT "localhost:9200/movies" -H 'Content-Type: application/json' -d'
{
  "settings": {
    "index": {
      "number_of_shards": 3,
      "number_of_replicas": 1
    }
  }
}'
```

- **Explication :**

  Cet exercice permet aux étudiants de créer un index avec un nombre défini de shards primaires et de replicas, leur permettant de voir comment Elasticsearch distribue les données pour optimiser les performances et la tolérance aux pannes.

Exercice 27 : Simulation d'une Panne

- **Tâche :** Observer le comportement du cluster après l'arrêt d'un nœud.
- **Action :**

- Arrêtez un des nœuds Elasticsearch.
- Vérifiez l'état du cluster avec la commande suivante :

```
curl -XGET 'localhost:9200/_cluster/health?pretty'
```

- **Explication :**

  Cet exercice aide les étudiants à comprendre comment Elasticsearch gère la perte d'un nœud et comment les shards répliqués assurent la continuité des opérations.

Exercice 28 : Gestion de l'Élasticité du Cluster

- **Tâche :** Augmentez la capacité du cluster en ajoutant des nœuds et observez le rééquilibrage des shards.
- **Action :**
  - Lancez un ou plusieurs nouveaux nœuds Elasticsearch sur votre système.

  - Modifiez la configuration pour augmenter le nombre de replicas :

```
curl -X PUT "localhost:9200/movies/_settings" -H 'Content-Type:
application/json' -d'
{
  "index": {
    "number_of_replicas": 2
  }
}'
```

- Vérifiez l'état du rééquilibrage des shards avec :

```
curl -XGET 'localhost:9200/_cat/shards?pretty'
```

- **Explication :**

  Cet exercice montre comment Elasticsearch redistribue les données à travers les nœuds supplémentaires pour améliorer la performance et la résilience du cluster.

Exercice 29 : Administration et Diagnostic de Cluster

- **Tâche :** Apprendre à utiliser les outils d'administration pour surveiller et diagnostiquer l'état du cluster.
- **Commande :**

```
# Visualiser l'état de santé du cluster
curl -XGET 'localhost:9200/_cluster/health?pretty'

# Visualiser l'état détaillé des shards
curl -XGET 'localhost:9200/_cat/shards?pretty'
```

- **Explication :**

  Cet exercice donne aux étudiants les outils nécessaires pour surveiller l'état du cluster et identifier les éventuels problèmes de répartition des shards ou autres problèmes de configuration.