Elasticsearch

Travaux Pratiques

zenika
ARCHITECTURE INFORMATIQUE

TP1 Installation

Objectif

L'objectif de ce TP est d'installer Elasticsearch sur un poste de travail (Windows ou Unix), ainsi que le plugin "Elasticsearch Head" qui sera utile pour la suite des travaux pratiques.

Note : L'API Rest/JSON d'Elasticsearch nécessite l'installation d'un client Rest. Ce client peut-être le plugin "Elasticsearch Head", cUrl, RestConsole ou un autre outil similaire. Vous pouvez utiliser l'outil qui vous convient le mieux pour réaliser les Tps.

Les commandes à exécuter dans un Shell (Invite de commandes Windows ou shell Bash) sont écrites avec la police de caractères Courier New

Préparation du poste de travail

- Vérifier la présence de Java sur le poste de travail

```
java -version
```

- Installer si besoin Java et positionner la variable d'environnement `JAVA_HOME`
- Créer un dossier `formation` à la racine dans lequel vont se trouver toutes les ressources de TP (binaires, application de démo, fichiers json, etc.). Les archives de Elasticsearch et Kibana devront être extraites dans ce dossier.

Installer Elasticsearch

- Récupérer une distribution d'Elasticsearch auprès du formateur ou directement sur le site officiel https://www.elastic.co/downloads
- Installer Elasticsearch
- Vérifier la présence des répertoires `bin` et `config`
- Editer le fichier `elasticsearch.yml` et configurez les propriétés suivantes
    - `cluster.name : formation-<votre trigramme>`
    - `node.name : noeud0`
- Démarrer Elasticsearch
- Vérifier dans le fichier de logs que le démarrage est réussi
- Vérifier la présence de nouveaux répertoires
    - A quoi correspondent-ils ?

- Stopper Elasticsearch

Installer le plugin Elasticsearch-Head

Elasticsearch-Head est un plugin pour Elasticsearch qui fournit une interface simple pour exécuter des requêtes Rest/JSON. Ce plugin est détaillé dans le chapitre `11 – Fonctionnalités Avancées` de la formation, mais il peut être très pratique pour la suite des travaux pratiques.

- Vérifier que Elasticsearch est arrêté
- Si Internet est accessible depuis le poste de travail,
  - Installer Elasticsearch-Head en exécutant la commande suivante:

```
bin\plugin install mobz/elasticsearch-head
```

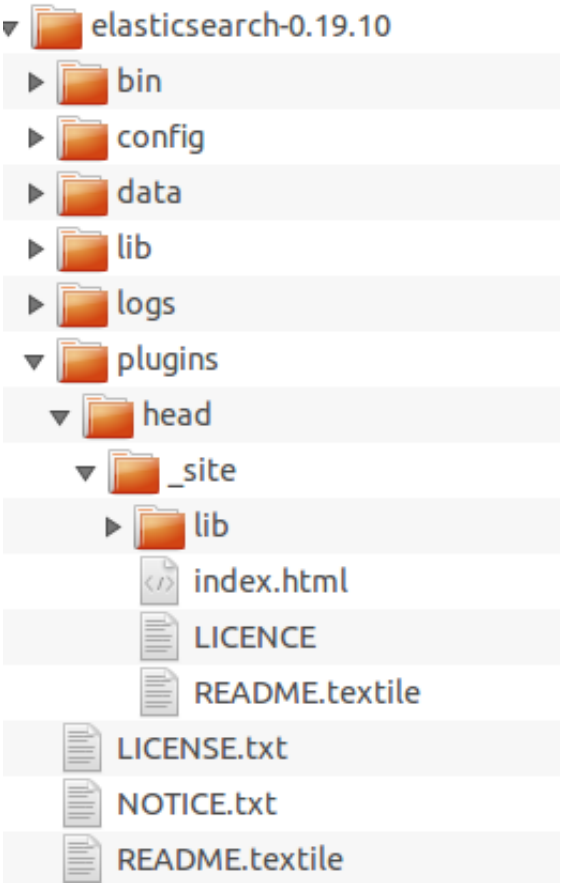
- Si Internet n'est pas accessible depuis votre poste de travail :
  - Récupérer une distribution d'Elasticsearch-Head auprès du formateur

```
bin\plugin install file:///C:/chemin/vers/mobz/elasticsearch-head-master
.zip
```

- Démarrer Elasticsearch
- Vérifier que le plugin est bien installé
  - La ligne suivante doit apparaître dans le fichier de log

```
[plugins] [noeud1] loaded [], sites [head]
```

- Lancer un navigateur Web et consulter la page d'Elasticsearch-Head http://localhost:9200/_plugin/head/
- La structure des répertoires doit être la suivante



Indexer des documents

- Vérifier que Elasticsearch est démarré
- Indexer dans l'index `zenika` un premier document de type `produit` d'identifiant `1` ayant les propriétés suivantes
  - code : S10_1949
  - nom : 1952 Alpine Renault 1300
  - echelle : 1:10
  - fournisseur : Classic Metal Creations
  - stock : 7305
  - prix : 98.58
  - devise : EUR

Note : ce document sera souvent réindexé au cours du TP, il peut être utile de copier/coller le document JSON dans un fichier texte.

- Quelle est la version de ce document dans Elasticsearch ?
- Récupérer le document indexé avec la commande

```
GET /zenika/produit/1
```

- Quel est le contenu du champ `_source` de la réponse ?
- Indexer dans le même index un second document d'identifiant `2` ayant les propriétés suivantes
  - code : S24_3371
  - nom : 1971 Alpine Renault 1600s
  - echelle : 1:24
  - fournisseur : Welly Diecast Productions
  - stock : 7995
  - prix : 38.58
  - devise : EUR

Rechercher des documents

- Rechercher tous les documents contenant le mot `Renault`
  - Combien y a t'il de résultats ?
  - Quelle est leur pertinence ?
- Rechercher tous les documents contenant le mot `EUR`
  - Combien y a t'il de résultats ?
- Rechercher tous les documents dont le champ `fournisseur` contient le mot `classic`
  - Combien y a t'il de résultats ?
- Dans Elasticsearch-head, quelle est la couleur de l'indicateur de santé du cluster ? Il doit normalement être jaune

Installer Kibana et Sense

- Extraire l'archive Kibana. Cette opération peut prendre un certain temps du fait du grand nombre de fichiers.
- Ouvrir une console dans le dossier Kibana
- Installer le plugin Sense dans Kibana

```
bin\kibana plugin -i elastic/sense
```

- Si l'accès à internet pose problème:
  - Récupérer une archive du plugin sense
  - Installer le plugin depuis l'archive

```
bin\kibana plugin -i elastic/sense -u file:///C:/chemin/vers/sense-latest.tar.gz
```

- Démarrer Kibana

```
bin\kibana
```

- Aller dans Sense http://localhost:5601/app/sense
- Essayer l'autocomplétion, exécuter une requête

```
POST /zenika/_search
{
  "quer..."
}
```

TP2 Indexation de Documents

Objectif

Dans ce TP, nous allons installer un cluster avec 2 nœuds puis créer des index. Nous verrons alors comment les index et les documents sont distribués sur les nœuds.

Installer un deuxième nœud Elasticsearch

- Installer un autre nœud Elasticsearch dont le nom sera `noeud1` en suivant la procédure du TP 1.3 Attention, le cluster.name de ce nouveau nœud doit être identique à celui de `noeud0`
- Démarrer le noeud1
- Attendre quelques secondes, puis vérifier dans Elasticsearch-head la couleur de l'indicateur de santé du cluster. Il doit désormais être vert
- Combien comptez-vous de shards et de replicas pour l'index `zenika` ?
- Comment sont-ils répartis sur les nœuds ?

Gestion des index

- Exécuter une requête pour supprimer l'index `zenika`
- Créer un index `data0` avec 2 shards et aucun replica
    - Comment sont distribués les shards sur les nœuds ?
    - Quel est l'indicateur de santé du cluster ?
- Modifier les settings de l'index `data0` pour ajouter 1 replica
    - Comment sont distribués les shards ?
    - Quel est l'indicateur de santé du cluster ?
    - A votre avis, quelle serait la distribution et la santé du cluster si l'on avait replica = 2 ?

Indexation et mise à jour de documents

- Ré indexer le premier document du TP 1.5 dans l'index `data0` avec une méthode POST
    - Quelles sont l'identifiant et le numéro de version du document indexé?
- Mettre à jour le prix de ce document en le réindexant entièrement

    - Quel est le nouveau numéro de version ?
- Mettre à jour le prix de ce document avec l'API `Update`
    - Note : depuis la version 1.2, il peut être nécessaire de configurer script.disable_dynamic: false dans le fichier de configuration et relancer les nœuds Elasticsearch

Versioning

- Ré indexer le second document du TP 1.5 dans l'index `data0` avec une méthode PUT et l'identifiant `2`
- Vérifier que son numéro de version est égal à 1
- Essayer de mettre à jour le document avec la commande suivante

```
PUT /data0/produit/2?version=2
{ "code" : "007" }
```

- Le document est il correctement mis à jour ?
- Que signifie l'erreur obtenue ?

```
?VersionConflictEngineException[[data0][1] [produit][2]: version conflic
t, current [1], provided [2]]; ","status":409}
```

(Facultatif) Alias d'index

- Créer un nouvel index `data1` avec shard = 2 et replica = 1
- Indexer dans l'index `data1` un document d'identifiant `1` ayant les propriétés suivantes
    - code : S24_3191
    - nom : 1969 Chevrolet Camaro Z28
    - echelle : 1:24
    - fournisseur : Exoto Designs
    - stock : 4695
    - prix : 50.51
    - devise : EUR

- Créer un alias d'index appelé `data` et qui pointe vers les index `data0` et `data1`
- Rechercher tous les documents dont le champ `devise` est `EUR` dans l'index `data0`, puis `data1` et enfin dans l'alias `data`
    - Combien y a t'il de résultats à chaque fois?
- Indexer un document dans l'alias d'index `data`. Que se passe-t-il?

TP3 Mappings

Objectif

Dans ce TP, nous utilisons le cluster (2 nœuds) pour configurer de nouveaux mappings.

Suppression des index

- Supprimer tous les index et documents

```
DELETE /_all
```

Ajout de mapping

- Créer dans l'index `data` un mapping pour le type de document `produit` ayant les spécificités suivantes
  - Les champs `code`, `echelle` et `devise` sont de type string non analysé
  - Le champ `fournisseur` est de type string analysé, il a pour valeur par défaut "Inconnu"
  - Le champ `nom` est de type multiple avec un champ par défaut de type string analysé et un champ `original` de type string non analysé
  - Le champ `stock` et `prix` sont de type numérique
  - Seul le champ `nom` est inclus dans le champ `_all`
  - Les champs `nom`, `code` et `fournisseur` sont stockés.
- Vérifier que le mapping a bien été créé

```
GET /data/_mapping
```

- Ré-Indexer à nouveau le premier document du TP 1.5
- Rechercher tous les documents contenant le mot `Classic`
  - Combien y a t'il de résultats ?
- Rechercher tous les documents contenant le mot `Classic` dans le champ `fournisseur`
  - Combien y a t'il de résultats ?
- Modifier le mapping pour que le champ `fournisseur` soit inclu dans le champ

`_all`

- Ré exécuter les recherches précédentes (sur "Classic")
  - Que constatez vous ?
- Ré-Indexer à nouveau le premier document du TP 1.5 puis refaite une recherche sur `Classic`
  - Que constatez vous ?
- Essayer de modifier le mapping pour changer le type de champ `prix` en string
  - Que constatez vous ?

TP4 Analyseurs

Objectif

Dans ce TP, nous allons tester certains analyseurs et modifier les mappings en conséquence.

API Analyze

- Utiliser l'API Analyze pour tester la chaîne de caractères "1:10" avec différents tokenizers :
  - whitespace
  - standard
  - keyword
  - letter

```
GET _analyze?pretty=true&tokenizer=whitespace&text=1:10
```

- Quel sont les résultats obtenus ?
- Tester la chaîne de caractères "1952 Alpine Renault 1300" avec le tokenizer `keyword` et différents token filters :
  - lowercase
  - reverse
  - ngram
- Tester la chaîne de caractères "Les àccènts sont nôs âmïs" avec l'analyseur `standard`
  - Quel est le résultat ?
- Quelle combinaison de tokenizer et de token filters conseilleriez-vous pour obtenir le résultat suivant ?
  - les accents sont nos amis

(Facultatif) Modification des mappings

- Avec le mapping construit en TP 3.3, quel est l'analyseur utilisé lors de l'indexation d'un champ `code` ?
- Quelle est la recherche à effectuer pour retrouver le produit de code "S10_1949" ?

- Quel mapping et analyseurs utiliser pour pouvoir rechercher un produit à partir d'un fragment de son code (ex : `s10` ou `1949`) ?

```
GET /data/produit/_search?q=code:s10
```

- Tester le mapping

TP5 Recherches & Filtres

Objectif

Dans ce TP, nous allons effectuer des recherches plus complexes et utiliser la notion de filtre.

Préparation du jeu de données

- Récupérer les fichiers `index-produits.json` et `produits.json` auprès du formateur
- Créer l'index en utilisant le fichier `index-produits.json` fourni

```
PUT /produits
{ ...index-produits.json... }
```

- Indexer les documents en utilisant le fichier `produits.json` fourni

```
POST /produits/_bulk
{ ...produits.json... }
```

- Vérifier le nombre de documents existants dans l'index `produits`

```
GET /produits/produit/_count
```

- Il doit y avoir 110 documents

Exécuter des requêtes de recherches

- Maintenant que les produits sont indexés, à vous de tester les recherches vues pendant le cours ! Voici tout de même quelques suggestions :
  - Recherche match_all
  - Recherche query_string sur les mot-clés `wheel`, `Germany`, `rotating wheels`, `rotating AND wheels`
  - Recherche query_string sur le champ "fournisseur.pays:france" et "fournisseur.pays:France"
  - Recherche query_string sur le champ "fournisseur.nom:auto*"
  - Recherche match sur le champ `description` et les mot-clés `rotating wheels`, `rotating AND wheels`

    - Rechercher les produits de devise `EUR`
    - Rechercher les produits dont le pays du fournisseur commence par `Un`
    - Rechercher les produits dont l'année est comprise entre 1951 et 1969 et d'origine France, trié par année croissante et pertinence
    - Rechercher les produits avec une année située aux alentours de 1951
    - Rechercher et récupérer uniquement les code de produits
    - ...

Utiliser les Filtres

- Combiner les recherches précédentes avec des filtres. Voici des suggestions de filtres :
    - Filtrer les produits sur une année (1971)
    - Filtrer les produits par échelle (1:24) et type (Planes)
    - Rechercher tous les produits qui n'ont pas d’année renseignée
    - Rechercher tous les produits présents en stock à plus de 5000 exemplaires
    - Rechercher tous les produits dont le prix est compris entre 25 et 50 euros
    - Utiliser la pagination des résultats from / size
    - ...

TP6 Agrégations

Objectif

Dans ce TP, nous allons utiliser les agrégations sur le jeu de données indexé au TP5.

Utiliser les agrégations

- Dénombrer tous les produits par pays
  - France: 29
  - United States : 25
  - ...
- Calculer le prix moyen par pays
- Calculer la valeur du stock minimum par pays, en triant les pays par prix moyen décroissant (noter la valeur du stock)
- Récupérer le code du produit avec le moins de stock pour chaque pays
- (Facultatif) Par devise, puis par décennie (`"interval": "3650d"`), calculer le prix moyen
- (Facultatif) Par devise, calculer quelle décennie a connu le prix moyen le plus faible

TP7 Recherche avancée

Objectif

Dans ce TP nous allons mettre en œuvre des fonctionnalités de recherche avancée comme le calcul de score, l'auto-completion et le highlighting.

Vous serez plusieurs fois amenés à corriger le mapping et ré-indexer les données. Pour cela:

1. Supprimer l'index

```
DELETE produits
```

2. Re-créer l'index

```
PUT produits
{ ... index-produits.json ...}
```

3. Re-remplir l'index

```
POST _bulk
{ ... produits.json ...}
```

Pertinence ou Score

1. En utilisant un requête de type `match` sur le champ `nom`, rechercher les produits contenant `Ford Model T`

- Comparer le score des différents résultats
- Activer l'explication du score et analyser le résultat

2. L'utilisateur préfère les produits fabriqués en France. Favoriser les résultats ayant un fournisseur français.

Auto-completion ou Suggestion

1. Corriger le mapping pour ajouter un champ `nom.completion` de type `completion` en utilisant le multi-fields.
2. Ré-indexer les données en utilisant le mapping corrigé
3. Essayer d'auto-compléter avec l'API Suggest

- `For`, `ford mod`
- `Model`

Highlighting

1. Corriger le mapping pour ajouter

- Sur le champ `nom` : `index_options: offsets`
- Sur le champ `description` : `term_vector: with_positions_offsets`

2. Ré-indexer les données en utilisant le mapping corrigé
3. En utilisant une requête `match_phrase` sur le champ `description`, rechercher les documents contenant `steering wheel` et mettre en relief les données trouvées.
4. En utilisant une requête `multi_match` de type `cross_fields` sur les champs `nom` et `description`, rechercher les documents contenant `Ford steering system` et mettre en relief les données trouvées.
5. Observer le Term Vector stocké pour le premier document:

```
GET produits/produit/1/_termvectors
```

TP8 Utiliser l'API Java

Objectif

Dans ce TP, nous allons lancer Eclipse et construire un petit projet Java permettant de communiquer avec notre cluster Elasticsearch.

Préparation du projet Java

- Lancer Eclipse et créer un nouveau projet Java (ou Maven) `formation`
- Récupérer la librairie elasticsearch auprès du formateur
- Importer cette librairie en tant que dépendance dans votre projet Java

Utilisation de l'API Java

Le reste du TP consiste à utiliser l'API Java pour communiquer avec le cluster et effectuer des recherches.

- Dans Eclipse, créer le package `com.zenika.training`
- Créer une classe Java EsClient avec une méthode main
- Instancier un Node pour communiquer avec le cluster Elasticsearch
- Récupérer le document d'identifiant "50" avec les méthodes `prepareGet()`, `execute()` et `actionGet()`
- Afficher dans le console (`System.out`) le résultat obtenu
- Lancer la classe EsClient et vérifier que la communication s'effectue correctement
- Modifier la classe EsClient pour ajouter une recherche `query_string` sur les mots clés "engine" et "metal". Il faut utiliser la méthode `prepareSearch()`.

Pour aller plus loin, vous pouvez :

- Utiliser des filtres et des agrégations
- Créer un nouvel index
- Vérifier l'existence d'un index
- Supprimer des documents
- Utiliser un `TransportClient` au lieu d'un `Node`
- ...

zenika ARCHITECTURE INFORMATIQUE

Aide mémoire

Index

```
PUT bibliotheque
{ "settings": {
    "index": {
      "number_of_shards": 5, "number_of_replicas": 1
    },
    "analysis": { ... }},
  "mappings": {
    "livre": { ... }}}

PUT bibliotheque/_settings
{ "index": {
    "number_of_replicas": 1 }}

DELETE bibliotheque

GET _cluster/health?level=indices
GET _all/_stats/docs,store?human
```

Mappings

```
PUT bibliotheque/_mapping/livre
{ "properties": {
    "titre": {
      "type": "string",
      "index": "analyzed", "analyzer": "french",
      "fields": {
        "original": {
          "type": "string",
          "index": "not_analyzed" }}},
    "prix": {
      "type": "double", "include_in_all": false },
    "date_parution": {
      "type": "date", "format": "YYYY-MM-dd" }}}

GET bibliotheque/_mapping
```

Documents

```
PUT bibliotheque/livre/123
{ ... }

POST bibliotheque/livre
{ ... }

GET bibliotheque/livre/123

DELETE bibliotheque/livre/123
{ ... }

POST bibliotheque/livre/123
{ "doc": {
    "prix": 19.99 }}

POST bibliotheque/livre/123
{ "script": "ctxt._source.stock -= 1" }
```

Analyse

```
POST /zenika/
{ "settings": {
    "analysis": {
      "analyzer": {
        "mon_analyzer": {
          "type": "custom",
          "tokenizer": "mon_tokenizer",
          "filter": ["lowercase", "mon_token_filter"],
          "char_filter": ["mon_char_filter"]}},
      "tokenizer": {
        "mon_tokenizer": { ... }},
      "filter": {
        "mon_token_filter": {  ... }},
      "char_filter": {
        "mon_char_filter": { ... }}}}}

GET _analyze?analyzer=french&text=Bonjour tout le monde

GET bibliotheque/_analyze?field=titre&text=Le crime de l'Orient Express
```

Recherche

```
GET bibliotheque/_search?q=Misérable&size=100&sort=titre:asc

POST bibliotheque/_search
{ "query": {
    "match": {
      "titre": "Misérables" }},
  "sort": [{"titre.lowercase":"asc"}],
  "size": 100, "from":10,
  "aggs": {
    "par_prix": {
      "histogram": {
        "field": "prix", "interval":10 }}}
}
```

Alias

```
PUT /bibliotheque/_alias/ma_bibliotheque

DELETE /bibliotheque/_alias/ma_bibliotheque

POST /_aliases
{ "actions": [
    { "remove":
      { "index": "bibliotheque", "alias": "ma_bibliotheque" }},
    { "add":
      { "index": "bibliotheque_v2", "alias": "ma_bibliotheque" }}]}
```