Elasticsearch

Premiers pas avec Elasticsearch

Mapping, Shard et Replica

- Un **mapping**
  - Est configuré au niveau d'un index
  - Définit la structure des champs d'un type de document
- Un **index** est divisé en shards
  - Un shard est une partition d'un index plus volumineux
  - Le nombre de shards est fixé à la création de l'index
- Les **shards** peuvent avoir des replicas
  - Un replica est une réplique (ou copie) d’un shard
  - Le nombre de replicas est configurable « à chaud »

Installation 1/3

- Elasticsearch nécessite un JRE ≥ 8u20 ou ≥ 7u55
  - Variable d'environnement `JAVA_HOME` doit être positionnée
- Distribué sous forme d'archive tar.gz, zip, package RPM, DEB standalone ou des repos fournis par Elasticsearch
- Démarrage d'une instance d'Elasticsearch avec la commande
  - `./bin/elasticsearch` (Unix)
  - `./bin/elasticsearch.bat` (Windows)
- Aucune configuration requise pour un 1er lancement

```
wget  https://download.elastic.co/.../elasticsearch-2.4.1.tar.gz
tar  -xzvf  elasticsearch-2.4.1.tar.gz
cd  elasticsearch-2.4.1/
./bin/elasticsearch
```

Installation 2/3

- Variables d'environnement spécifiques :
  - `ES_JAVA_OPTS` : options spécifiques de la JVM d'Elasticsearch
  - `ES_HEAP_SIZE` et `ES_MAX_MEM` : taille de la heap de la JVM
    - Affecter environ 50 % de la RAM disponible
- Exécuter Elasticsearch avec les options de la JVM
  - `./bin/elasticsearch -Xmx2g -Xms2g -d`
  - `./bin/elasticsearch --node.name=noeud1 --cluster.name=production`

Installation 3/3

- Linux: utiliser les paquets DEB ou RPM
- Windows: utiliser la commande `service.bat`:

```
bin\service.bat  install  elasticsearch_2
bin\service.bat  stop  elasticsearch_2
```

- Voir https://www.elastic.co/guide/en/elasticsearch/reference/current/_installation.html

Arborescence des répertoires

Répertoire	Description	Chemin par défaut	Configuration
home	Répertoire de base d'Elasticsearch		path.home
bin	Regroupe les exécutables	{path.home}/bin	
conf	Fichiers de configuration	{path.home}/conf	path.conf
data	Données d'indexation. Peut contenir les données de plusieurs noeuds	{path.home}/data	path.data
work	Fichiers temporaires	{path.home}/work	path.work
logs	Logs du serveur	{path.home}/logs	path.logs

- Les chemins sont tous configurables :
  - Avec un fichier de configuration

```
path.data:  /mnt/data
```

  - En paramètre de l'exécutable

```
elasticsearch  -Des.path.data=/mnt/data
```

Configuration d'un nœud

- Elasticsearch possède deux principaux fichiers de configuration
  - `elasticsearch.yml` et `logging.yml`
  - Situés dans le répertoire `config`
  - Par défaut formaté en YAML (.yml), peut être remplacé par du JSON

```
# ======================== Elasticsearch Configuration =========================
...
# Please see the documentation for further information on configuration options:
# <http://www.elastic.co/guide/en/elasticsearch/reference/current/setup-
configuration.html>
# ---------------------------------- Cluster -----------------------------------
# Use a descriptive name for your cluster:
# cluster.name: my-application
# ------------------------------------ Node ------------------------------------
# Use a descriptive name for the node:
# node.name: node-1
# ...
# ---------------------------------- Paths ------------------------------------
# Path to directory where to store the data (separate multiple locations by
comma):
# path.data: /path/to/data
# Path to log files:



# path.logs: /path/to/logs
```

Plugins 1/2

- Système de plugins intégré (binaire dédié `plugin`)
- Installer un plugin manuellement :
  - Télécharger le plugin
  - Copier le plugin dans le répertoire « plugins » d'Elasticsearch

```
$ ./bin/plugin install <org>/<user/component>/<version>
$ ./bin/plugin install file:///chemin/du/plugin.zip
```

- Exemple :

```
$ ./bin/plugin install mobz/elasticsearch-head
-> Installing mobz/elasticsearch-head...
Trying https://github.com/mobz/elasticsearch-head/archive/master.zip ...
Downloading .....DONE
Verifying https://github.com/mobz/elasticsearch-head/archive/master.zip
checksums if available ...
NOTE: Unable to verify checksum for downloaded plugin (unable to find .sha1 or
.md5 file to verify)
Installed head into /chemin/vers/elasticsearch-2.4.1/plugins/head
```

Plugins 2/2

- Nombreux types de plugins : Alerting, Analysis, Discovery, Management and Site, Mapper, Scripting, Security, Backup, Transport
- Principales interfaces Web (Management and Site plugins):

Nom	Installation	Type de support	Licence	Description
Head	mobz/elasticsearchhead	Communauté	Gratuit	Plugin de visualisation "historique"
Kopf	lmenezes/elasticsearch kopf	Communauté	Gratuit	Equivalent de Head plus "évolué"

Communiquer avec Elasticsearch 1/2

- Il existe plusieurs façons de communiquer avec Elasticsearch
- **L'interface Rest**
  - Propose toutes les fonctionnalités d'Elasticsearch
  - Utilisable quelque soit la technologie/langage du client
  - https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html
  - Très utilisé, même en production
- **Le client Java**
  - Permet de se connecter à un cluster Elasticsearch
  - https://www.elastic.co/guide/en/elasticsearch/client/java/api/current/index.html
  - L'api interne d'Elasticsearch

Communiquer avec Elasticsearch 2/2

- Les autres clients et frontends
  - Librairies tierces proposant des clients Elasticsearch pour de nombreux langages : php, perl, scala, python...
  - Interfaces de monitoring ou d'administration
  - https://www.elastic.co/guide/en/elasticsearch/client/index.html

Vue générale de l'API Rest

- API Rest

```
http://host:port/[index]/[type]/[_action|id]
```

- Permet de réaliser toutes les opérations possibles
- Les requêtes et les réponses sont formatées en JSON
  - `index` : Nom de l'index sur lequel porte l'opération
  - `type`: Nom du type de document
  - `_action`: Nom de l'action à effectuer
  - `id` : Identifiant du document
  - **Méthode**: Dépend du genre d'opération à réaliser sur la ressource: `GET, POST, PUT, DELETE`

Outillage Rest

- En ligne de commande
  - cUrl

```
curl -XGET "http://localhost:9200/bibliotheque/_search?pretty=true"

curl -XPOST "http://localhost:9200/bibliotheque/bibliotheque/_search" -d '
{ "query": { ... }}'

curl -XPUT "http://localhost:9200/bibliotheque/bibliotheque/1" --data-
binary @livre1.json
```

  - HTTPie
- Plugins navigateur
  - Chrome: Postman, Advanced Rest Client, Insomnia
  - Firefox: Rest Client, Rest Easy
- Plugins Elasticsearch: Head, Kopf...
- Plugin Kibana: Sense

Première indexation d'un document 1/2

- Une API REST / JSON permet de communiquer avec Elasticsearch, par exemple pour indexer un document :

```
PUT  /bibliotheque/livre/1
{  "titre"  :  "Spring  Batch  in  Action",
   "auteurs"  :  [
      {"prenom"  :  "Arnaud",  "nom"  :  "Cogoluegnes"},
      {"prenom"  :  "Thierry",  "nom"  :  "Templier"},
      {"prenom"  :  "Gary",  "nom"  :  "Gregory"}],
   "prix"  :  59.99,  "devise"  :  "USD",
   "publication"  :  "2011-10","langue"  :  "en_US"}
```

  - `bibliotheque` est le nom de l'index dans lequel le document sera stocké
  - `livre` est le type du document
  - `1` est l'identifiant du document

Première indexation d'un document 2/2

- Le document est fourni au format JSON

```
PUT /bibliotheque/livre/1
{ "titre" : "Spring Batch in Action",
  "auteurs" : [
     {"prenom" : "Arnaud", "nom" : "Cogoluegnes"},
     {"prenom" : "Thierry", "nom" : "Templier"},
     {"prenom" : "Gary", "nom" : "Gregory"}],
  "prix" : 59.99, "devise" : "USD",
  "publication" : "2011-10","langue" : "en_US"}
```

- Elasticsearch répond dans ce même format

```
{ "_index": "bibliotheque", "_type": "livre", "_id": "1", "_version": 1,
  "_shards": { "total": 2, "successful": 1, "failed": 0 },
  "created": true
}
```

Récupération d'un document

- L'API REST/JSON permet de récupérer un document dès lors qu'on connaît son type et son identifiant :

```
GET /bibliotheque/livre/1
```

```
{ "_id": "1",
  "_index": "bibliotheque",
  "_type": "livre",
  "_version": 1,
  "found": true,
  "_source": {
      "titre" : "Spring Batch in Action",
      ...
      "langue" : "en_US"}}
```

Première recherche 1/2

- Recherche de tous les livres qui contiennent « action »

```
GET  /bibliotheque/livre/_search?q=action
```


```
{  "took"  :  103,  "timed_out"  :  false,
   "_shards"  :  {  "total"  :  5,  "successful"  :  5,  "failed"  :  0},
   "hits"  :  {"total"  :  1,  "max_score"  :  0.054244425,
      "hits"  :  [
         {"_index"  :  "bibliotheque","_type"  :  "livre","_id"  :  "1",
            "_score"  :  0.054244425,
            "_source"  :  {
               "titre":"Spring  Batch  in  Action",
               "auteurs":[
                  {"prenom":"Arnaud","nom":"Cogoluegnes"},
                  {"prenom":"Thierry","nom":"Templier"},
{"prenom":"Gary","nom":"Gregory"}
               ],
               "prix":59.99,"devise":"USD",
               "publication":"201110","langue":"en_US"}}
      ]}}
```

Première recherche 2/2

- `hits.total` indique le nombre total de résultats trouvés
- Pour chaque résultat ou `hit` :
  - Ses coordonnées : l'`index` auquel il appartient, le `type` de document, et l'identifiant unique
  - La `source` du document
  - La pertinence du document ou `score`

Recherche sur plusieurs types

- Indexation d'un document de type `dvd` dans l'index `bibliotheque`

```
PUT  /bibliotheque/dvd/1
{  "titre"  :  "Les  Looney  Tunes  passent  à  l'action",
    "acteurs"  :  [
        {"prenom"  :  "Timothy",  "nom"  :  "Dalton"},
        {"prenom"  :  "Jenna",    "nom"  :  "Elfman"},
        {"prenom"  :  "Brendan",  "nom"  :  "Fraser"}
    ],
    "sortie"  :  "2003-12"}
```

Recherche sur plusieurs types

- Recherche des livres et dvd contenant « action » dans le titre

```
GET  /bibliotheque/livre,dvd/_search?q=titre:action
```

- Recherche sur tous les types de documents contenant « action » dans le titre

```
GET  /bibliotheque/_search?q=titre:action
```

- Recherche sur plusieurs champs

```
GET  /bibliotheque/_search?q=auteurs.nom:templier  OR  acteurs.nom:dalton
GET  /bibliotheque/_search?q=auteurs.\*:templier
GET  /bibliotheque/_search?q=\*.nom:d*
```

Recherche sur plusieurs index

- Recherche des livres et dvd sur plusieurs index

```
GET  /bibliotheque1,bibliotheque2/livre,dvd/_search?q=titre:action
```

- Recherche sur tous les index (`_all`)

```
GET  /_all/_search?q=titre:action
```

- Recherche sur tous les index commençant par « lib » sauf « library »

```
GET  /+lib*,-library/_search?q=venus
```