Elasticsearch

Plan

- Introduction aux moteurs de recherche
- Premiers pas avec Elasticsearch
- Indexation de documents
- Mapping
- Analyse et extraction de texte
- Recherche de documents
- Agrégations
- Fonctionnalités avancées de recherche
- Clients et API Java
- Cloud et clusterisation
- Fonctionnalités avancées

Introduction aux moteurs de recherche

Définition

« Un moteur de recherche est une application permettant de retrouver des ressources associées à des mots quelconques » Wikipedia

Moteurs de recherche | Web spécialisés

- Santé, Alimentation,
- Actualités, Informations, Presse,
- Voyages, Transports,
- Cartographie, Blogs, Réseaux
- sociaux, Forums, Livres, Média,
- Vidéos, Photos, Services Publics,
- Shopping, ...

Moteurs de recherche | Éditeurs

- Sinequa
- Dassault Systems Exalead
- Antidot
- Celebros
- Oracle Endeca
- FredHopper
- ...

Solutions open source

- Apache Lucene
- Apache SolR
- Elasticsearch
- Hibernate Search
- Compass
- Sphinx
- ...

Le besoin

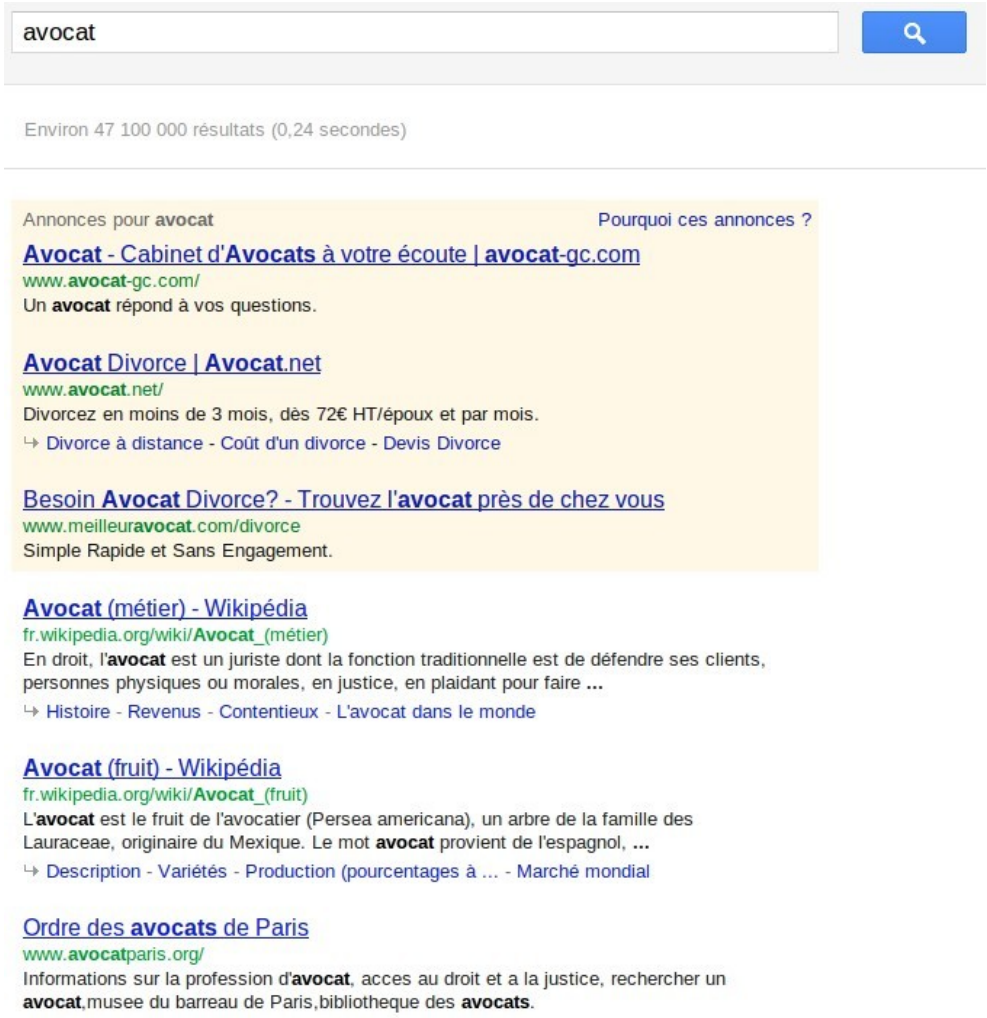
- Pourquoi avoir besoin d'un moteur de recherche ?

Le besoin

Le besoin

Les fonctionnalités clés

- Pertinence (ou Scoring)
  - La pertinence des résultats est le « nerf de la guerre »
  - Formule de calcul sur laquelle il est possible d'intervenir
    - Boost
    - Top hits / Always on top



Les fonctionnalités clés

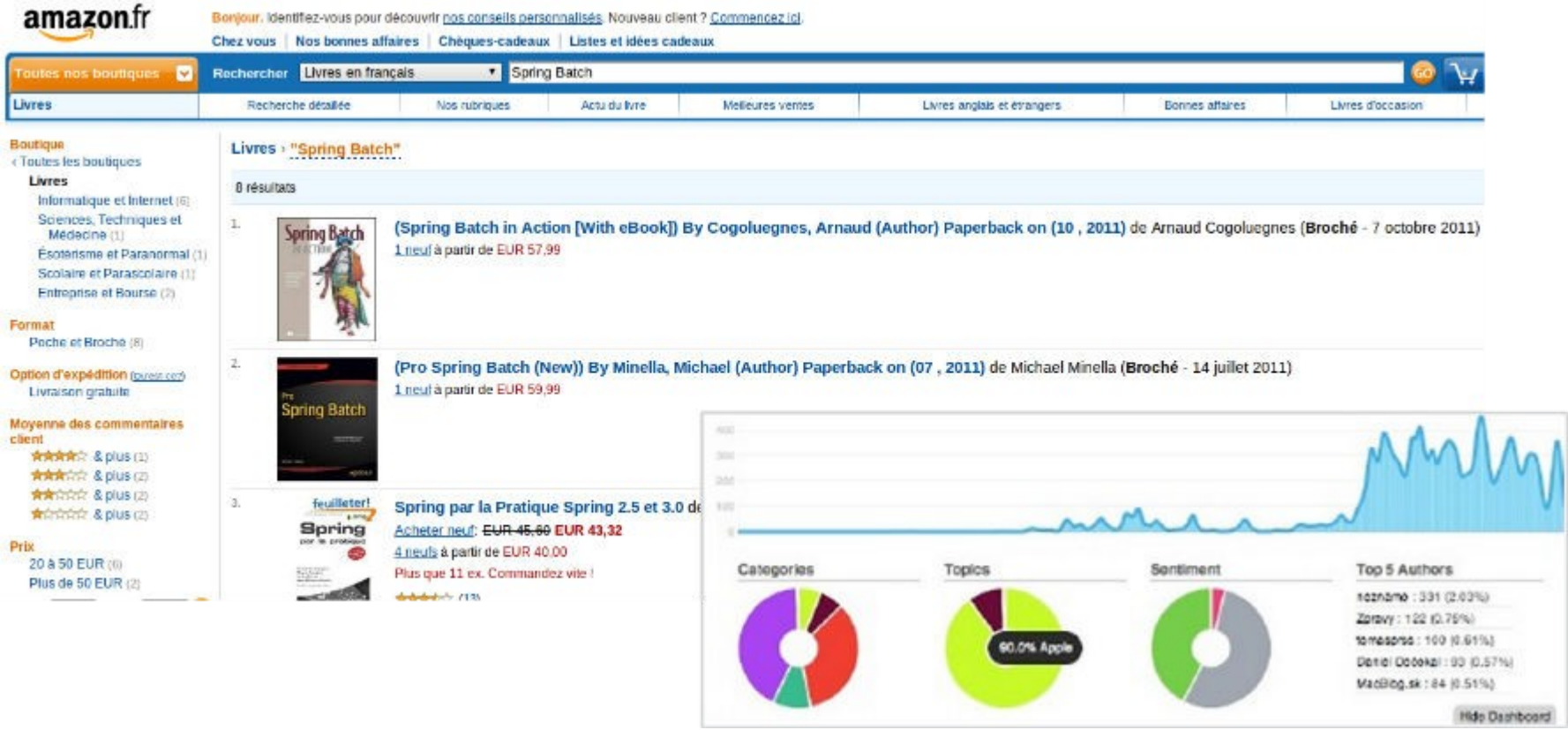
- Autosuggestion (ou AutoSuggest)
  - Faire gagner du temps à l'utilisateur
  - Éviter une recherche sur des termes mal orthographiés
  - Orienter les recherches vers des motsclés populaires ou privilégiés
  - Réutiliser des résultats mis en cache

Les fonctionnalités clés

- Correction orthographique
  - Faire gagner du temps à l'utilisateur
  - Éviter une recherche sur des termes mal orthographiés
  - Mise en œuvre complexe : nécessite l'utilisation de thésaurus et de dictionnaires spécifiques à chaque langue
  - Alternative basée sur la recherche approximative et les synonymes

Les fonctionnalités clés

- Facettes (ou Facets) et agrégations
  - Permettent de « catégoriser » et analyser des résultats de recherche
  - Permettent d'affiner une recherche



Les fonctionnalités clés

- More Like This
  - Lancer une nouvelle recherche à partir d'un résultat de recherche
  - Permet de trouver des résultats semblables à un 1er résultat
  - Incite et oriente l'utilisateur

Les perspectives

- Orientation sociale
  - Google « +1 »
- Personnalisation des résultats
  - Machine learning : le moteur de recherche « apprend » des recherches effectuées par l'utilisateur
- Effet de « bulle »
- Utilisation et fonctionnement dans le Cloud

Analyse et résumé des résultats

Mise en œuvre d'un moteur de recherche

Mise en œuvre d'un moteur de recherche

- Quelle que soit la solution technique choisie, la mise en place d'un moteur de recherche nécessite :
  - La collecte et l'indexation des données
  - La recherche et restitution des résultats

Collecte et Indexation des données

- Les différentes étapes de l'indexation des données

Collecte du contenu

- Identifier et récupérer les informations à indexer
- Origine des données très variée : système de fichiers, base de données, messagerie, système de gestion de contenus, sites web...
- Complexité supplémentaire si des droits d'accès spécifiques sont nécessaires pour accéder à l'information (« superadmin », ACLs)
- Volumétrie des données pouvant nécessiter une indexation incrémentale et/ou « live »
- Outils d'indexation spécifiques : crawlers, connecteurs ou batchs
  - SolR, Nutch, Grub, Apache Droids, Heritrix, Aperture, ManifoldCF...

Construction des documents

- Transformation du contenu en document compréhensible par le moteur de recherche
- Un document est constitué d'un ou plusieurs champs, chaque champ a un format et une ou plusieurs valeurs
- La structure des documents à indexer doit être conçue en fonction des futures recherches et de l'affichage des résultats
- A ce stade, certaines opérations peuvent être réalisées :
  - Extraction de texte
  - Boosting
  - Uniformisation des valeurs, retraitements
  - Association de documents

Exemples de documents

- Exemples de documents de type « Livre »

Identifiant	Champs	Valeur(s)
1	titre	Spring Batch in Action
	auteurs	Arnaud Cogoluegnes, Thierry Templier, Gary Gregory
	prix	59.99
	devise	USD
	publication	10/2011
	langue	en_US
2	titre	Mahout in Action
	auteurs	Sean Owen, Robin Anil
	prix	44.99
	devise	USD
	publication	10/2011
	langue	en_US

Exemples de documents

Identifiant	Champs	Valeur(s)
3	titre	L'art en théorie et en action
	auteurs	Nelson Goodman
	prix	6.60
	devise	EUR
	publication	01/2009
	langue	fr_FR
4	titre	Spring par la pratique
	auteurs	Arnaud Cogoluegnes, Thierry Templier, Julien Dubois
	prix	43.30
	devise	EUR
	publication	10/2011
	langue	fr_FR

Exemple de document pour elasticsearch

- Exemple de document au format JSON pour elasticsearch

```
{ "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"}
```

Analyse des documents

- Les documents sont analysés par le moteur de recherche
- Le moteur de recherche analyse les champs de chaque document et applique éventuellement des traitements
- Ces traitements permettent d'obtenir le comportement souhaité lors des recherches :
  - Recherches non sensibles à la casse
  - Gestion des synonymes
  - Recherche phonétique
  - Gestion des pluriels et des conjugaisons
  - ...

Exemple d'analyse réalisée par elasticsearch

- Texte original

Spring Batch in Action

Mahout in Action

L'art en théorie et en action

Spring par la pratique

- Texte après analyse

spring batch action

mahout action

art theorie action

spring pratique

Indexation des documents

- Lors de cette phase, les documents sont ajoutés à l'index du moteur de recherche
- L'organisation interne de l'index est spécifique à chaque moteur de recherche
- De nombreux moteurs de recherche utilisent un « Inverted Vector Index »
  - Chaque mot/token référence les documents où il est présent
  - Recherche rapide mais indexation plus longue
- Les traitements réalisés lors de l'analyse peuvent impacter la taille de l'index

Exemple d'indexation de documents 1/2

- Texte après analyse

spring batch action

mahout action

art theorie action

spring pratique

- Termes uniques

spring, batch,
action, mahout,
art, theorie,
pratique

Exemple d'indexation de documents 2/2

- Termes uniques

spring, batch, action, mahout, art, theorie, pratique

- Construction de l'index

	spring	batch	action	mahout	art	theorie	pratique
Spring Batch in Action	X	X	X				
Mahout in Action			X	X			
L'art en théorie et en action			X		X	X	
Spring par la pratique	X						X

Recherche et restitution des résultats

- Les différentes étapes de la recherche de documents

Formulaire de recherche

- Le formulaire de recherche permet de retrouver des documents à partir de motclés et d'options de recherche
- Il peut aussi permettre de filtrer / d'affiner la recherche
- Une attention particulière doit être apportée à l'interface graphique de recherche en privilégiant la simplicité

Requête de recherche

- A partir des données sélectionnées dans le formulaire de recherche, l'application doit construire une requête de recherche
- A ce stade, des filtres supplémentaires peuvent être ajoutés automatiquement
  - Par exemple, pour filtrer les résultats en fonction des habilitations de l'utilisateur
- La requête est ensuite exécutée dans le moteur de recherche, qui calcule éventuellement une **pertinence / scoring**
- Les recherches les plus courantes peuvent être mises en cache si besoin

Exemple de recherche

- Recherche de « spring action »

	spring	batch	action	mahout	art	theorie	pratique
Spring Batch in Action	O	X	O				
Mahout in Action			O	X			
L'art en théorie et en action			O		X	X	
Spring par la pratique	O						X

Pertinence

- Un indice de pertinence de chaque résultat est éventuellement calculé par le moteur, c'est le **scoring** (ou relevance)
- Pour calculer la pertinence :
  - Combien de termes recherchés le document contientil ?
  - Combien de fois le terme recherché estil présent dans le document ?
  - Quelle est la fréquence d'apparition de chaque terme dans tous les documents ?
- La pertinence peut être personnalisée :
  - Par document et par champ
- Lors de l'indexation ou lors de la recherche

Restitution des résultats

- Présente la liste des résultats de recherche fournis par le moteur de recherche
- L'interface graphique peut permettre à l'utilisateur d'affiner les résultats (facettes, filtres...)
- Ne pas masquer les traitements effectués (correction orthographique, boost, filtres...)
- Utilisez votre propre application !

Solutions de recherche open source

Apache Lucene

- Projet open source développé en Java et commencé en 2000 par
  - Doug Cutting (aussi à l'origine des projets Nutch, Hadoop)
  - Erik Hatcher (Ant, Tapestry)
  - Otis Gospodnetic (Lucene, Hadoop)
- Version actuelle 6.3.0 disponible sur http://lucene.apache.org/ Fournit
- un moteur de recherche et un ensemble de classes utilitaires
  - Classes d'analyse et découpage de texte
  - Constructeurs de requêtes de recherche
  - Constructeurs de documents à indexer
  - Gestionnaires d'index
- Lucene se retrouve au cœur de nombreux autres moteurs de recherche
  - API bas niveau (filesystem, structure des documents indexés)
  - Utilisation complexe en environnement multithreadé

Apache SolR

- Projet open source développé en Java
  - Première release (1.3) en 2006
  - Fusionné avec Apache Lucene en
- 2011
- Basé sur la librairie Apache Lucene
- Version actuelle 6.3.0 disponible sur http://lucene.apache.org/solr/
- Développement conjoint avec Lucene
  - Application Web (WAR) fournissant un serveur de moteur de
  - recherche API HTTP/XML, JSON
  - Recherche distribuée
  - Interface d'administration
  - Réplication maître/esclave
  - Support des facettes, géolocalisation
  - Configuration par fichiers XML
- Outils d'import de données

Elasticsearch

- Projet open source développé en Java
  - Première release en 2010
  - Développé par Shay Banon (à l'origine de Compass)
- Basé sur la librairie Apache Lucene et sur l'expérience acquise sur le projet Compass
- Version actuelle 2.4.1 disponible sur https://www.elastic.co/downloads/elasticsearch
- Serveur de moteur de recherche capable de fonctionner en cluster
  - API REST/JSON
- Recherche distribuée en mode cluster
  - Taillé pour de gros volumes de
  - données Configuration en JSON et
  - YAML Structure de données nonfigée
  - Recherche multiindex
- Support des facettes, géolocalisation

Premiers pas avec Elasticsearch

Elasticsearch

- Elasticsearch est un **moteur de recherche** développé en Java
  - Open source (Licence Apache 2), Distribué, RESTful
- Basé sur la librairie Apache Lucene, il en masque la complexité
- Elasticsearch fournit des **services Web**
  - Communication possible avec une API HTTP/REST et JSON
- Elasticsearch fonctionne en **cluster**
  - Un cluster est un ensemble d'instances d'Elasticsearch
  - Une instance d'Elasticsearch est appelée « node » ou « noeud »
  - Améliore les **performances** et la **disponibilité** du moteur de recherche

Fonctionnement en Cluster

- Un **cluster** Elasticsearch
  - Est identifié par un nom unique
  - Est composé de nœuds situés sur un même réseau
  - Élit un seul nœud maître
- Un **node** Elasticsearch
  - Est un processus Java standard
  - Est toujours rattaché à un cluster
  - Stocke tout ou partie des données indexées
  - Peut être ajouté ou retiré du cluster à chaud
  - Peut proposer des services Web HTTP
  - REST/JSON
  - Communique avec les autres nœuds pour répondre aux requêtes pour tout le cluster
  - Répond aux requêtes des clients

Document, Type et Index

- Un **document**
  - Représente les données métier à indexer et sur lesquelles des recherches seront effectuées
  - Formaté en JSON

```
{ "titre" : "Spring Batch in Action",
  "auteurs" : ["Arnaud Cogoluegnes", "Olivier Bazoud"],
  "prix" : 20.80}
```

- Un **type de document**
  - Regroupe les documents de même type
- Un **index**
  - Est un espace de stockage des documents dont les types sont fonctionnellement liés

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
```

Conl,guration 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.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:9
   200/bibliotheque/_s
   earch?pretty=true"

curl -XPOST
   "http://localhost:9
   200/bibliotheque/bi
   bliotheque/_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

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":"Cogolue
            gnes"},
            {"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
{ "acteurs" : [
    {"prenom" : "Timothy", "nom" : "Dalton"},
    {"prenom" : "Jenna", "nom" : "Elfman"},
    {"prenom" : "Brendan", "nom" : "Fraser"}
```

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
```

TP1

Indexation de documents

Index | Notion de shard & replica 1/3

- Les documents JSON sont typés et stockés dans un index
- Un **Index** est constitué de shards et de replicas
- **Shard**
  - Un shard est un fragment de l'index
  - L'ensemble des shards constitue l'index
- **Replica**
  - Un replica est une copie de l'index
  - Les replicas assurent les sauvegardes de l'index en cas de crash des nœuds
- Par défaut, les shards et replicas sont automatiquement répartis sur les différents nœuds du cluster Elasticsearch

Index | Notion de shard & replica 2/3

- Configuration par défaut dans le fichier `elasticsearch.yml` :
  - `index.number_of_shards: 5`
  - `index.number_of_replicas: 1`
- Configuration pour chaque index lors de la création de l'index
  - Le nombre de shards est fixé **lors de la création de l'index**
  - Le nombre de replicas peut être **diminué/augmenté à chaud**

Index | Notion de shard & replica 3/3

- **Plus de shards** améliore les performances à l'indexation et permet de distribuer un index volumineux sur plusieurs nœuds
- **Plus de replicas** améliore les performances à la recherche et la disponibilité de l'index
- La configuration par défaut convient dans de nombreux cas, mais seuls des tests de performance permettent de trouver le bon nombre de shards/replicas à utiliser

Création d'un index

- Création d'un index avec l'API REST « Create Index » :

```
PUT /bibliotheque/
```

- Création d'un index avec des settings :

```
PUT /bibliotheque/
{ "settings" : {"index" : {
    "number_of_shards" : 8,
"number_of_replicas": 2
}}}
```

- Les index sont automatiquement créés s'ils n'existent pas au moment de l'indexation des documents

Modification d'un index

- Mise à jour des **settings** d'un index avec l'API REST «Update Settings » :

```
PUT /bibliotheque/_settings
{ "index" : {
    "number_of_replicas": 6,
    "refresh_interval": "30s"
}}
```

- Principaux settings :
  - `number_of_replicas`
  - : Nombre de réplicas

    `refresh_interval` : Intervalle de temps (en secondes) de rafraîchissement de l'index

Suppression d'un index

- Suppression d'un index avec l'API REST :

```
DELETE /bibliotheque/
```

- Suppression de plusieurs index

```
DELETE /index1,index2,index3/
```

- Suppression de tous les index

```
 DELETE
/_all DELETE
 /*
```

- Il est possible d'empêcher la suppression de tous les index en configurant dans le fichier `elasticsearch.yml`:

```
action.destructive_requires_name : true
```

Indexation de documents avec l'API Rest

- L'API REST permet d'indexer un document au format JSON
  - En utilisant la méthode HTTP PUT et en précisant un identifiant de document :

```
PUT /bibliotheque/livre/1
{"titre": "7 ans après...", "auteurs": "Guillaume Musso", "prix": 20.80}
```

```
{ "created":true, "_index":"bibliotheque", "_type":"books", "_id":"1", "_version":1,
 "_shards": { ...          }}
```

- En utilisant la méthode HTTP POST pour qu'un identifiant de document soit automatiquement généré :

```
POST /bibliotheque/livre
{"titre": "7 ans après...", "auteurs": "Guillaume Musso", "prix": 20.80}
```

```
{ "created":true, "_index":"bibliotheque", "_type":"books",
"_id":"AVGRV_usnjDD0CQltR_N", "_version":1,
   "_shards": { ...          }}
```

Mise à jour d'un document 1/3

- Mettre à jour un document consiste à **réindexer le document**
  - Nécessite de connaître son identifiant
- Incrémente le numéro de version automatiquement
- Annule et **remplace** la version précédente du
  - document
  - Nécessite de réindexer la totalité du document

```
PUT /bibliotheque/livre/1
{ "titre": "7 ans après.....", "prix": 20.80}
```

Mise à jour d'un document 2/3

- L'API REST « Update » permet de mettre à jour un ou plusieurs champs d'un document via script
- Permet :
  - d'ajouter de nouveaux champs,
  - d'ajouter de nouvelles valeurs à un champ existant,
  - de supprimer un champ existant,
  - supprimer le document si une condition est vérifiée...
- Comme indiqué, réindexer totalement le document et nécessite donc que la `_source` du document soit activée.
- Utilise en interne la `version` du document pour détecter les accès concurrents (voir plus loin pour plus de détails sur le fontionnement du champ version)

Mise à jour d'un document 3/3

Deux syntaxes :

- A base de `script` :

```
POST /bibliotheque/livre/1/_update
  { "script" : "ctx._source.prix += addition",
    "params" : {"addition" : 4}}
```

  - map `ctx` permet d'accéder à plusieurs
  - infos (`_index`, `_type`, `_id` ...)
  - nécessite d'autoriser l'exécution de scripts: `script.inline: true`
  - suppression de champs possible:

```
"script" : "ctx._source.remove(\"prix\")"
```

- A base de document partiel `doc` :

```
POST /bibliotheque/livre/1/_update
{ "doc" : {"prix" : 25 }}
```

  - mergé avec le document actuel
  - pas de suppression possible

Suppression d'un document

- L'API REST permet de supprimer un document

```
DELETE /bibliotheque/livre/1
```

API Bulk 1/3

- Endpoint `_bulk` dédié aux opérations en lot

```
POST /_bulk
{ ... }
```

- Beaucoup plus performante pour un grand nombre d'opérations
  - Exemple: Batch
- Format spécifique des données :

```
action + meta_donnees\n
source_optionelle\n
action + meta_donnees\n
source_optionelle\n
...
```

- En sortie : JSON contenant le résultat de chaque action
  - code réponse HTTP
  - éventuellement, message d'erreur

API Bulk 2/3

- Opérations possibles : `index` / `create` / `update` / `delete`
- Source inutile pour `delete`
- Exemple de données en entrée :

```
{ "create" : { "_index" : "bibliotheque", "_type" : "livre", "_id" : "43" } }
{ "auteur" : "Victor Hugo", "titre":"Les Misérables" }
{ "delete" : { "_index" : "bibliotheque", "_type" : "livre", "_id" : "2" } }
{ "update" : {"_index" : "bibliotheque", "_type" : "dvd", "_id" : "1"} }
{ "doc" : {"lang" : "fr"} }
...
```

API Bulk 3/3

- Réponse :


```
[{ "create": {
      "_index": "bibliotheque",
      "_type": "livre",
      "_id": "43",
      "_version": 1,
      "status": 201 }
},{ "delete": {
      "_index": "bibliotheque",
      "_type": "livre",
      "_id": "2",
      "_version": 1,
      "status": 404, "found":
      false }
},{ "update": {
      "_index": "bibliotheque",
      "_type": "dvd",
      "_id": "43",
      "status": 404, "error": {
            "type":
            "document_missing_exception",
            "reason": "[dvd][1]: document
            missing", "index": "bibliotheque",
            "shard": "-1" }}
},...]
```

Identifant de document

- Tous les documents ont un identifiant
  - Lors de l'indexation du document, l'identifiant peut être spécifié (`PUT`) ou généré automatiquement (`POST`)
  - L'**identifiant et le type de document** permettent d'identifier un document dans un index
  - L'id est retourné dans le champ `_id` lors d'une recherche
  - L'id est utilisé pour récupérer un document dans une requête `GET`
  - Le champ `_uid` est l'identifiant unique dans l'index `= _type + _id`
- Utiliser autant que possible un id métier
  - Toutes les opérations sur un document passent par l'identifiant
  - Sans l'identifiant, il faut préalablement faire une recherche pour retrouver un document

Version

- Les documents sont automatiquement **versionnés** lors de l'indexation
  - Le numéro de version commence à 1
  - Il est incrémenté à chaque nouvelle indexation du document
  - Le numéro de version est retourné lors de l'indexation mais pas lors des recherches (par défaut)
- Il est possible de gérer manuellement les numéros de version
  - En précisant les paramètres version et `version_type=external`

Version | Optimistic Concurrency Control 1/4

- **Optimistic Concurrency Control**
  - Elasticsearch n'a **pas de support de transactions**
  - Lors d'une indexation, il est possible de fournir un numéro de version via le paramètre `version`
  - Elasticsearch vérifiera si le document est dans la version indiquée avant d'effectuer l'opération demandée
  - Utile pour simuler une transaction de type readthenupdate

Version | Optimistic Concurrency Control 2/4

- **Optimistic Concurrency Control**
  - Exemples de requêtes illustrant l'Optimistic Concurrency Control :
  - https://www.elastic.co/blog/versioning

Version | Optimistic Concurrency Control 3/4

- Question ? Expliquez

```
version_conflict_
engine_exception)
```

```
{"error":{"reason":
 "[livre][1]: version
 conflict, current [4],
 provided [3]"
... }}, "status":409}
```

Version | Optimistic Concurrency Control 4/4

- Exemple de requête illustrant le `version_type=external`
  - Impose `version` > version stockée dans l'index

```
PUT /bibliotheque/livre/1?version_type=external&version=4
{"titre": "Spring Batch in Action"}
{"_index": "bibliotheque", "_type": "livre", "_id": "1", "_version": 4,
"created": false}
```

Refresh

- **Near Real Time**
  - Il existe un léger décalage entre le moment où un document est indexé et le moment où il est disponible et remonte en résultat de recherche
  - Ce laps de temps est configuré à 1 seconde par défaut pour les index
  - Configurable au niveau de chaque index avec le setting `refresh_interval`
- Avant une indexation en masse (`_bulk`), il est conseillé d'augmenter le `refresh_interval` pour éviter trop de rafraîchissements de l'index

Refresh et segments Lucene

- Avant d'être transmis à Lucene, les documents indéxés sont **analysés** et **stockés** dans un buffer mémoire:
  - à ce stade, ces documents **ne peuvent** pas être **recherchés**.
- Ce buffer mémoire est accompagné d'un fichier **TransLog** qui a le même cycle de vie et qui est écrit dans le **cache disque système**. Il sert à persister les commandes ayant rempli le buffer.
- Ce buffer mémoire est **flushé** dans un segment Lucene lorsque:
  - le buffer est **plein**
  - lors d'un **reopen** Lucene
  - sur un **refresh** ElasticSearch:
    - **sans commit** Lucene,
    - déclenché selon valeur du `index.refresh_interval` ou via l'API `_refresh`
  - sur un **flush** ElasticSearch:
    - provoque un **commit** Lucene,
      déclenché automatiquement par Elasticsearch ou via l'API `_flush`

Segments et merges Lucene 1/2

- Un segment Lucene est un **inverted vector index** autosuffisant et **immuable** (jamais de suppression)
- Les segments augmentent en nombre et **consomment** de plus en plus de **ressources** (pointeurs sur fichiers, caches, memoire...)
- Pour pallier ce problème, Elasticsearch choisit de **merger** régulièrement des segments Lucene
- Ces opérations de merge répondent à une **merge policy** choisie optionnellement à la création de l'index. (réglage **expert**)
- Un **merge** consiste à:
  - Supprimer plusieurs segments
  - Supprimer les caches associés
  - Construire un segment **immuable** plus grand pour contenir les segments précédents sans reprendre les documents **marqués** détruits

Segments et merges Lucene 2/2

- L'opération de merge peut être coûteuse, jouer sur les paramètres suivants au niveau du `noeud` pour maîtriser ce coût:
  - positionner `indices.store.throttle.type` to merge
  - positionner `indices.store.throttle.max_bytes_per_sec` à `x`mb

Voir cette simulation des merges Lucene pour comprendre: https://www.youtube.com/watch?v=YW0bOvLp72E

Refresh forcé

- Certaines opérations acceptent un paramètre `refresh=true` pour rafraîchir directement l'index après l'opération
  - Utile pour les tests d'intégrations
  - A utiliser judicieusement, car très coûteux

```
PUT /bibliotheque/livre/1?refresh=true
{"titre":"Spring Batch in Action"}
```

Operation Type

- Lors d'une indexation, il est possible de préciser le paramètre `op_type=create`
- Permet d'indexer un document uniquement s'il n'existe pas déjà

```
PUT /bibliotheque/livre/1?op_type=create
{"titre":"Spring Batch in Action"}
{ "error": {
  "type": "document_already_exists_exception",
  "reason": "[livre][1]: document already exists",
  "index": "bibliotheque", "shard": "2"
 }, "status": 409 }
```

- Peut aussi s'écrire avec la notation `_create`

```
PUT /bibliotheque/livre/1/_create
{"titre":"Spring Batch in Action"}
```

Autres fonctionnalités

- Lors d'une indexation de document, il est aussi possible de préciser
  - Le `routing` à utiliser, pour que le document soit ajouté à un shard précis de l'index
  - Le document `parent` du document qu'on s'apprête à indexer, et ainsi permettent les recherches parent/child
  - Le `consistency` à utiliser, pour avoir la main sur le nombre minimum de shards/replicas dans lequel le document doit être écrit: `one`, `quorum` (par defaut), `all`.
  - Un `timeout`, au delà duquel une erreur sera renvoyée si l'opération prend plus de temps à s'exécuter

Alias d'index

- Il est possible de créer des **alias** sur les index
  - Un alias pointe vers un ou plusieurs autres **index**
  - La création et la suppression des alias se fait via une API Rest

```
POST /_aliases
{"actions" : [{ "add" : { "index" : "bibliotheque", "alias" :      "znk" } }]}
PUT bibliotheque/_aliases/znk
GET bibliotheque/_aliases
```

  - Il existe un alias par défaut : `_all`
- Permet différents cas d'utilisations
  - Alias « current » sur le dernier index de logs
  - Réunir plusieurs index fréquemment utilisés ensemble
- Fonctionnalités avancées
  - Filtre : alias sur un index filtré sur une requête
  - Route : spécifie le routage sur les shards de l'index

TP2

Mapping