elasticsearch

elastic

Base de données NoSQL distribuée et Moteur de recherche

Fabien Silberstein

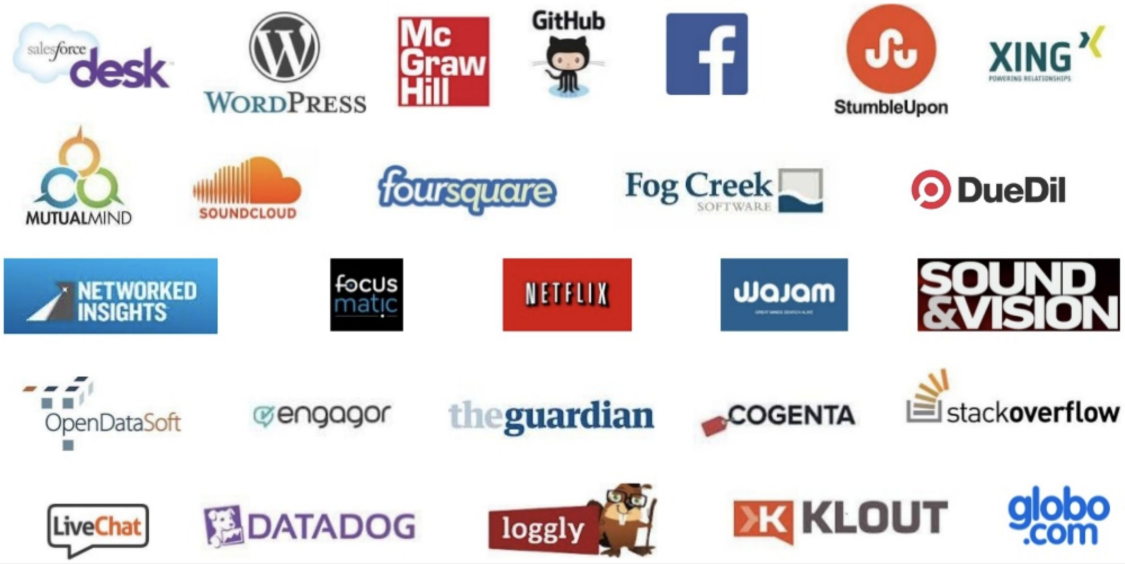
Quelques points clés

- Basé sur Lucene
- Développé en Java
- Schema en JSON
- Index et Recherche
- Sous License Apache (open source et gratuit)
- API REST
- Supporte la recherche par agrégation
- Distribué et prévu pour le Cloud
- Première version en 2010
- Versions supportées: 5 et 6
- Elastic Cloud, AWS, Azure, GCP, etc...

Clients

- cURL
- Java
- Javascript
- C#
- Python
- PHP
- Perl
- Ruby
- Et plus créés par la communauté...

Qui l'utilise ?



Concepts

Field

- Plus petite unité de données
- Typé: boolean, int, text, etc…
- Une collection de fields compose un document
- Le nom ne peut pas commencer par un caractère spécial et ne peut pas contenir de point

Concepts

Document

- Objet JSON
- Aucune limite sur le nombre de documents stockés dans un index
- Contient des champs clé-valeur
- Contient des champs réservés eg: _index, _type, _id

Concepts

Type

- Représente une “classe” unique de documents
- Défini par un **nom** et un **mapping**. Rempli le champ _type quand il est utilisé. Ce champ peut être utilisé pour filtrer les recherches.
- On peut définir autant de type que l’on veut dans chaque index

Concepts

Index

- Plus grande unité de données
- Partition logique de documents
- On peut aussi voir le nom “indice” dans ES
- Pas de limite sur le nombre d’index utilisés dans ES
- Contient des types, mappings, documents, champs

Concepts (comparés à une BDD SQL)

Elasticsearch	Base de données SQL
Index (Indice)	Base
Type	Table
Document	Ligne
Field	Colonne

Concepts

Mapping

- Définit le type de donnée pour chaque champ (string, int, etc…)
- Définit comment le champ doit être indexé et stocké
- Peut être défini explicitement
- Peut être généré automatiquement quand un document est indexé

Concepts

Shards

- Composant principal d'ES qui facilite sa "scalabilité". Peut être primaire ou duplicata
- Le nombre de shards par index est défini à la création de l'index et ne peut être modifié par la suite
- Un index est donc distribué à travers plusieurs shards. Cela permet la distribution des opérations à travers les noeuds pour améliorer les performances
- Chaque shard est un index Lucene indépendant et peut être hébergé n'importe où dans le cluster

Concepts

Replica

- Copie de chaque shard de votre index
- Système de sauvegarde utile quand un noeud tombe
- Utilisé pour les requêtes de type "Read", donc ajouter des replicas augmente la performance de recherche
- Comme les shards, le nombre de replica est défini à la création de l'index. Mais contrairement aux shards, ce nombre peut être modifié n'importe quand après la création de l'index

Concepts

Node

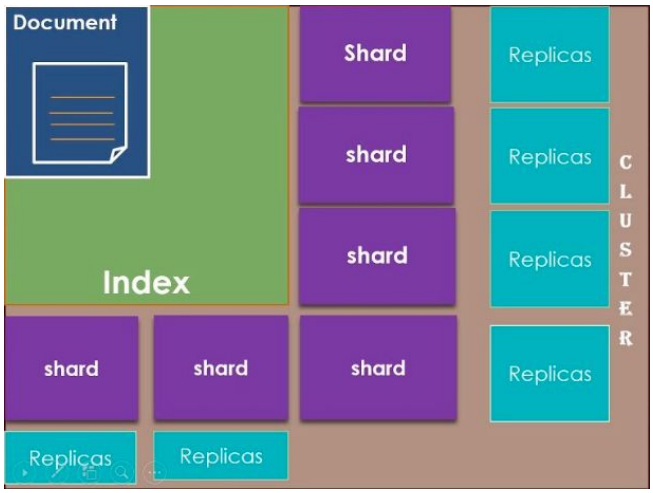
- Le coeur de toute installation ES. C'est en fait une **instance** ES.
- Par défaut, à chaque noeud est automatiquement assigné un identifiant unique, ou un nom, qui est utilisé pour les tâches d'administration. Cela devient encore plus important dans une installation multi-noeuds ou dans un environnement "clusterisé"

Concepts

Cluster

- Composé de un ou plusieurs noeuds. Comme les noeuds, un cluster possède un identifiant unique utilisé par les noeuds voulant joindre le cluster
- Dans chaque cluster, il y a un noeud master, qui est responsable des opérations au niveau du cluster (comme ajouter ou supprimer un noeud du cluster). Ce noeud est choisi automatiquement par le cluster, mais il peut changer en cas d'échec
- Quand le cluster grossit, il se réorganise automatiquement pour propager les données

Concepts

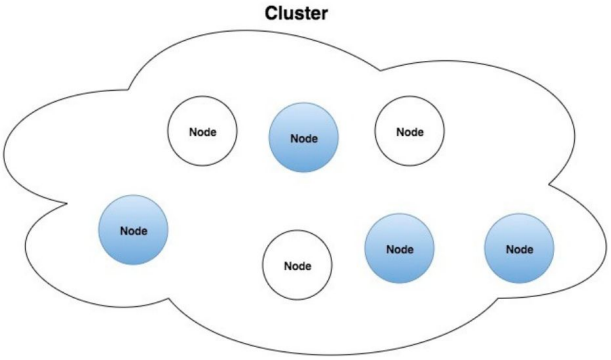


Scaling

- Vertical: plus de ressources hardware sur le même serveur
- Horizontal: plus de serveurs

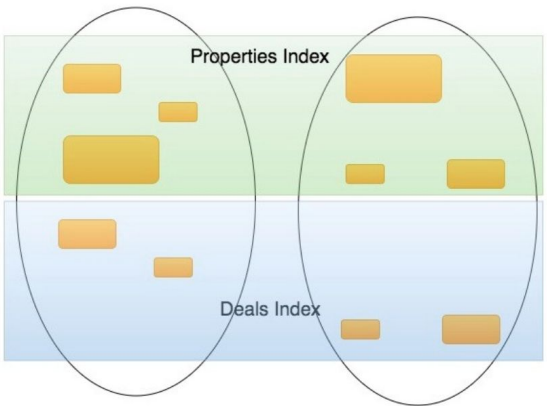
Horizontal Scaling

Un cluster ES n'est pas limité à une seule machine, on peut augmenter ce nombre à l'infini pour assurer un plus gros trafic et des données plus grosses.



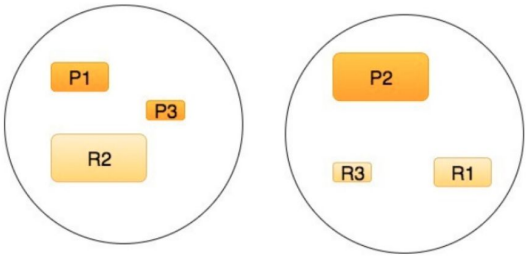
Horizontal Scaling

Chaque index est composé de shards à travers un ou plusieurs noeuds.



Ici, le cluster ES possède 2 noeuds, 2 indexes (*properties* et *deals*) et 5 shards dans chaque noeud.

Horizontal Scaling



Ecriture: seulement sur le primary

Lecture: tous les shards

Ici, nous avons 3 shards “primary” et 3 shards “replica”.

Les écritures (indexation) sont faite sur les shards primaires. Un shard peut avoir 0 ou plusieurs replica qui vont simplement copier les données.

Les shards primaires peuvent être sur plusieurs noeuds. Si un noeud tombe, les shards replica se trouvant sur un noeud fonctionnel seront élus primaire automatiquement.

Status des index

- “Green” - Tous les shards primaires sont disponibles et ils ont tous au moins un replica
- “Yellow” - Tous les shards primaires sont disponibles mais tous n’ont pas au moins un replica
- “Red” - Au moins un shard primaire est indisponible

```
$ curl 'http://localhost:9200/_cat/indices'
green open deals      5 1 265107 0   263mb 131.5mb
green open properties 5 1    301 0 532.9kb 266.4kb
```

Conclusion des concepts

- Les noeuds composent un cluster et contiennent des shards
- Les shards contiennent des documents que l'on recherche
- ES aiguille les requêtes à travers les noeuds
- Les noeuds fusionnent ensuite les résultats de chaque shard (index Lucene) pour créer un résultat de recherche

API REST

Vue générale

ElasticSearch propose une API REST permettant de réaliser tous types d'opérations. Cette API utilise le format JSON pour les requêtes et les réponses et supporte les principales méthodes HTTP (GET, DELETE, PUT et POST).

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

- 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

API REST
Première indexation

Voici un exemple d'indexation d'un document de type "cours" dans un index nommé "iut" avec l'API REST:

```
curl -XPUT 'http://localhost:9200/iut'

curl -XPUT 'http://localhost:9200/iut/cours/1' -d '{
    "heures": 2,
    "professeurs": [
        {
            "nom": "Silberstein",
            "prenom": "Fabien"
        }
    ],
    "sous-titre": "Comprendre et savoir utiliser ElasticSearch, un super moteur de recherche",
    "titre": "Cours ElasticSearch"
}'
```

Résultat renvoyé:

```
{
    "_id": "1",
    "_index": "iut",
    "_shards": {
        "failed": 0,
        "successful": 1,
        "total": 2
    },
    "_type": "cours",
    "_version": 1,
    "created": true,
    "result": "created"
}
```


API REST
Première indexation

L'indexation a généré un **type** "cours" automatiquement et l'a ajouté au **mapping** de l'index "iut"

```
curl -XGET 'http://localhost:9200/iut/_mappings?pretty'
{
  "iut" : {
    "mappings" : {
      "cours" : {
        "properties" : {
          "heures" : {
            "type" : "long"
          },
          "professeurs" : {
            "properties" : {
              "nom" : {
                "type" : "text",
                "fields" : {
                  "keyword" : {
                    "type" : "keyword",
                    "ignore_above" : 256
                  }
                }
              },
              "prenom" : {
                "type" : "text",
                "fields" : {
                  "keyword" : {
                    "type" : "keyword",
                    "ignore_above" : 256
                  }
                }
              }
            }
          },
          "sous-titre" : {
            "type" : "text",
            "fields" : {
              "keyword" : {
                "type" : "keyword",
                "ignore_above" : 256
              }
            }
          },
          "titre" : {
            "type" : "text",
            "fields" : {
              "keyword" : {
                "type" : "keyword",
                "ignore_above" : 256
              }
            }
          }
        }
      }
    }
  }
}
```

API REST

Récupérer un document par id

```
curl -XGET 'http://localhost:9200/iut/cours/1?pretty'
{
  "_index" : "iut",
  "_type" : "cours",
  "_id" : "1",
  "_version" : 1,
  "found" : true,
  "_source" : {
    ...
  }
}
```


API REST

Recherche simple par mot clé

```
curl -XGET 'http://localhost:9200/iut/cours/_search?q=elasticsearch&pretty'
{
  "took" : 37,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 0.38828257,
    "hits" : [
      {
        "_index" : "iut",
        "_type" : "cours",
        "_id" : "1",
        "_score" : 0.38828257,
        "_source" : {
            ...
        }
      }
    ]
  }
```

API REST

Recherche avancée

```
curl -XGET 'http://localhost:9200/iut/_search?pretty' -d '{
    "query": {
        "range": {
            "heures": {
                "gte": 1
            }
        }
    }
}'
```

Recherche tous les documents dans l'index "iut" qui ont un champ **"heures" >= à un**

```
curl -XGET 'http://localhost:9200/iut/_search?pretty' -d '{
    "query": {
        "bool" : {
            "must": [
                {
                    "range": {
                        "heures": {
                            "gte": 1
                        }
                    }
                },
                {
                    "term": {
                        "_type": "tp"
                    }
                }
            ]
        }
    }
}'
```

Recherche tous les documents dans l'index "iut" qui ont un champ **"heures" >= à un** ET qui sont de **"type" tp**

NB: ne renverra pas de résultat

```
{
  "took" : 7,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 0,
    "max_score" : null,
    "hits" : [ ]
  }
}
```


API REST

Tous les types de recherche ici:
https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html

- Query DSL
  - Query and filter context
  - Match All Query
  - Full text queries
    - Match Query
    - Match Phrase Query
    - Match Phrase Prefix Query
    - Multi Match Query
    - Common Terms Query
    - Query String Query
    - Simple Query String Query
- Term level queries
  - Term Query
  - Terms Query
  - Terms Set Query
  - Range Query
  - Exists Query
  - Prefix Query
  - Wildcard Query
  - Regexp Query
  - Fuzzy Query
  - Type Query
  - Ids Query

Et bien d'autres...

Analyse d'indexation

En reprenant le mapping du sous-titre:

```
46              "sous-titre" : {
47                "type" : "text",
48                "fields" : {
49                  "keyword" : {
50                    "type" : "keyword",
51                    "ignore_above" : 256
52                  }
53                }
54              },
```

Type par défaut: anciennement string et maintenant à la fois stocké comme "text" (full-text) et "keyword" (pour la recherche exact avec un champ nommé keyword)

```
curl -XGET 'http://localhost:9200/iut/cours/_search?pretty' -d '{
    "query": {
        "match": {
            "sous-titre": "comprendre"
        }
    }
}'

curl -XGET 'http://localhost:9200/iut/cours/_search?pretty' -d '{
    "query": {
        "match": {
            "sous-titre.keyword": "comprendre"
        }
    }
}'

curl -XGET 'http://localhost:9200/iut/cours/_search?pretty' -d '{
    "query": {
        "match": {
            "sous-titre.keyword": "Comprendre et savoir utiliser ElasticSearch, un super moteur de recherche"
        }
    }
}'
```

→ OK

→ Aucun resultat

→ OK

Analyse d'indexation

En reprenant le mapping du sous-titre:

```
46              "sous-titre" : {
47                "type" : "text",
48                "fields" : {
49                  "keyword" : {
50                    "type" : "keyword",
51                    "ignore_above" : 256
52                  }
53                }
54              },
```

Avec deux documents ayant pour sous-titre:

Id 1: "Comprendre et savoir utiliser ElasticSearch, un super moteur de recherche"

Id 2: "Comprendre elasticSearch"

On obtient la table d'indexation suivante:

Term	Document Id
comprendre	1,2
et	1
savoir	1
utiliser	1
elasticsearch	1,2
...	

Analyse d'indexation: Analyzer

En reprenant le mapping du sous-titre et en ajoutant un champ manquant avec sa valeur par défaut:

L'analyser est chargé d'examiner les données à indexer afin de les stocker de la façon la plus optimale pour les recherches.

Un analyzer se definit avec:

- Un **char filter** : applique une transformation sur le texte complet avant qu'il ne soit découpé en tokens
- Un **tokenizer** : coupe le texte en tokens
- Un **token filter** : peut ajouter, modifier ou supprimer des tokens

Analyse d'indexation: Analyzer

Définition du **Standard Analyzer**

Char filter: aucun

Tokenier:

- Standard Tokenizer: utilise le "Unicode Text Segmentation algorithm". Utilise l'espace comme séparateur ou les "-", etc...

Token filters:

- Lower Case Token Filter: modifie chaque token pour le passer en minuscules

Analyse d'indexation: Analyzer

Et si nous avions utilisé le **french Analyzer**

Char filter: aucun

Tokenier: le même que standard

Token filters:

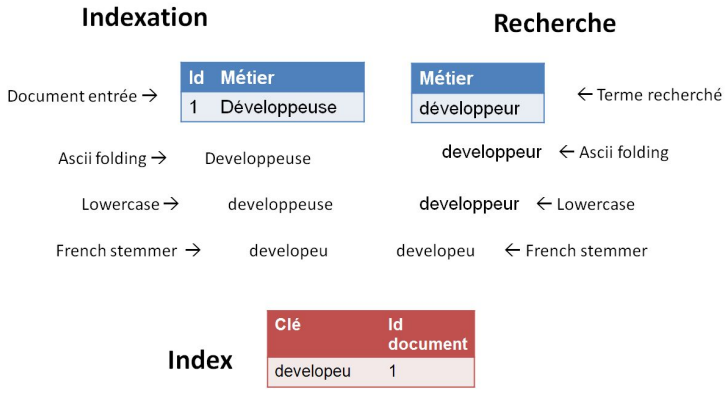
- french_elision: supprime les articles pouvant précéder un mot
- lowercase: comme le standard, met en minuscule l'intégralité du token
- french_stop: retire les tokens tels que "en", "au", "du", "par", "est", … car ils sont considérés comme du bruit
- french_stemmer: applique un racinisation (stemming) des tokens, c'est ce qui permet de supprimer les formes plurielles, les conjugaisons et accord de genre sur un mot

Définition complète de l'analyzer:
https://www.elastic.co/guide/en/elasticsearch/reference/6.x/analysis-lang-analyzer.html#french-analyzer

Analyse d'indexation: Analyzer

Que fait le stemmer du french analyzer sur l'indexation du document suivant?

```
curl -XPUT 'http://localhost:9200/iut/metier' -d '{
    "id": 1,
    "metier": "Développeuse"
}'
```



Analyse d’indexation: Analyzer

Différence sur le sous-titre

```
curl -XGET 'http://localhost:9200/_analyze?pretty' -d '{
  "analyzer" : "standard",
  "text" : "Comprendre et savoir utiliser ElasticSearch, un super moteur de recherche"
}'
{
  "tokens" : [
    {
      "token" : "comprendre",
    },
    {
      "token" : "et",
    },
    {
      "token" : "savoir",
    },
    {
      "token" : "utiliser",
    },
    {
      "token" : "elasticsearch",
    },
    {
      "token" : "un",
    },
    {
      "token" : "super",
    },
    {
      "token" : "moteur",
    },
    {
      "token" : "de",
    },
    {
      "token" : "recherche",
    }
  ]
}
```

```
curl -XGET 'http://localhost:9200/_analyze?pretty' -d '{
  "analyzer" : "french",
  "text" : "Comprendre et savoir utiliser ElasticSearch, un super moteur de recherche"
}'
{
  "tokens" : [
    {
      "token" : "comprendr"
    },
    {
      "token" : "savoi"
    },
    {
      "token" : "utilis"
    },
    {
      "token" : "elasticsearch"
    },
    {
      "token" : "sup"
    },
    {
      "token" : "moteu"
    },
    {
      "token" : "recherch"
    }
  ]
}
```


Analyse d’indexation: Analyzer

Attention le french analyzer n’est pas parfait!

Exemple:
Si un document est indexé avec le mot bœuf et recherché avec le mot boeuf, il ne sera pas retourné

On pourrait par exemple ajouter un nouveau tokenizer filter:
“french_synonym” qui permet de rechercher par synonymes aussi.

Analyse d'indexation: Analyzer

```
curl -XPUT 'http://localhost:9200/french_example' -d '{
  "settings": {
    "analysis": {
      "filter": {
        "french_elision": {
          "type":         "elision",
          "articles_case": true,
          "articles": [
              "l", "m", "t", "qu", "n", "s",
              "j", "d", "c", "jusqu", "quoiqu",
              "lorsqu", "puisqu"
            ]
        },
        "french_stop": {
          "type":       "stop",
          "stopwords":  "_french_"
        },
        "french_synonym": {
          "type": "synonym",
          "ignore_case": true,
          "expand": true,
          "synonyms": [
            "search, recherche"
          ]
        },
        "french_stemmer": {
          "type":       "stemmer",
          "language":   "light_french"
        }
      },
      "analyzer": {
        "rebuilt_french": {
          "tokenizer":  "standard",
          "filter": [
            "french_elision",
            "lowercase",
            "french_stop",
            "french_synonym",
            "french_stemmer"
          ]
        }
      }
    }
  }
}'
```

On peut tout à fait modifier / créer nos propre analyzer
Ici on ajoute un synonyme à recherche avec le mot anglais "search"

Maintenant une recherche comme celle-ci retournera le document:

```
curl -XGET 'http://localhost:9200/iut/cours/_search?pretty' -d '{
    "query": {
        "match": {
            "sous-titre": "search"
        }
    }
}'
```