NOSQL

Les bases de données

Au programme

JSON

MONGODB

NOSQL

ELASTICSEARCH

ElasticSearch

ElasticSearch

Présentation

- Moteur de recherche et d'analyse basé sur Lucene : « Je veux une recherche google sur mon application »
  - Indexation recherche et analyse de document JSON
  - Stockage de données distribuées
  - RestFul : Webservice Rest JSON
  - Intégration facile avec tous les langages
  - OpenSource publié sous licence Apache
  - Ecrit en Java
  - Première version sortie en février 2010
  - Très utilisé sur le web : Wikipédia, stackoverflow, github

  - Note : ElasticSearch est une base de données, mais il ne faut pas le considérer comme une base de données terminale

ElasticSearch

Différence avec les bases de données standards

Une base de données standard (SQL ou non) utilise des requêtes qualifiées de booleén : Soit la donnée satisfait la condition, soit elle n'est pas remontée.

ElasticSearch et les moteurs de recherche en général, permet de remonter des résultats qui correspondent totalement ou partiellement aux critères : Notion de pertinence, de score.

Repose sur trois critères :

1. Term Frenquency (TF) → la fréquence d'apparition des mots-clés de la recherche dans le document (les documents contenants le plus de « matches » sont priorisés)
2. Inverse document frequency (IDF) → un coefficient correcteur accordant plus d'importance aux mots-clés apparaissants dans peu de documents
3. Field-length norm → un autre coefficient correcteur portant sur la longueur du champ dans lequel le mot-clé est trouvé : ainsi, les matches dans le titre ou le nom du document ont un poids supérieurs aux matches dans le corps du document.

ElasticSearch

Version, maintenance et EOL

Version	Release date	Security Support (EOL)	Last Release
7.15	September 22, 2021	22 Mar 2023	7.15.2
7.14	August 03, 2021	03 Feb 2023	7.14.2
7.13	May 25, 2021	25 Nov 2022	7.13.4
7.12	March 23, 2021	23 Sep 2022	7.12.1
7.11	February 10, 2021	10 Aug 2022	7.11.2
7.10	November 11, 2020	11 May 2022	7.10.2
7.9	August 18, 2020	18 Feb 2022	7.9.3
7.8	June 18, 2020	18 Dec 2021	7.8.1
7.7	May 13, 2020	13 Nov 2021	7.7.1
7.6	February 11, 2020	11 Aug 2021	7.6.2

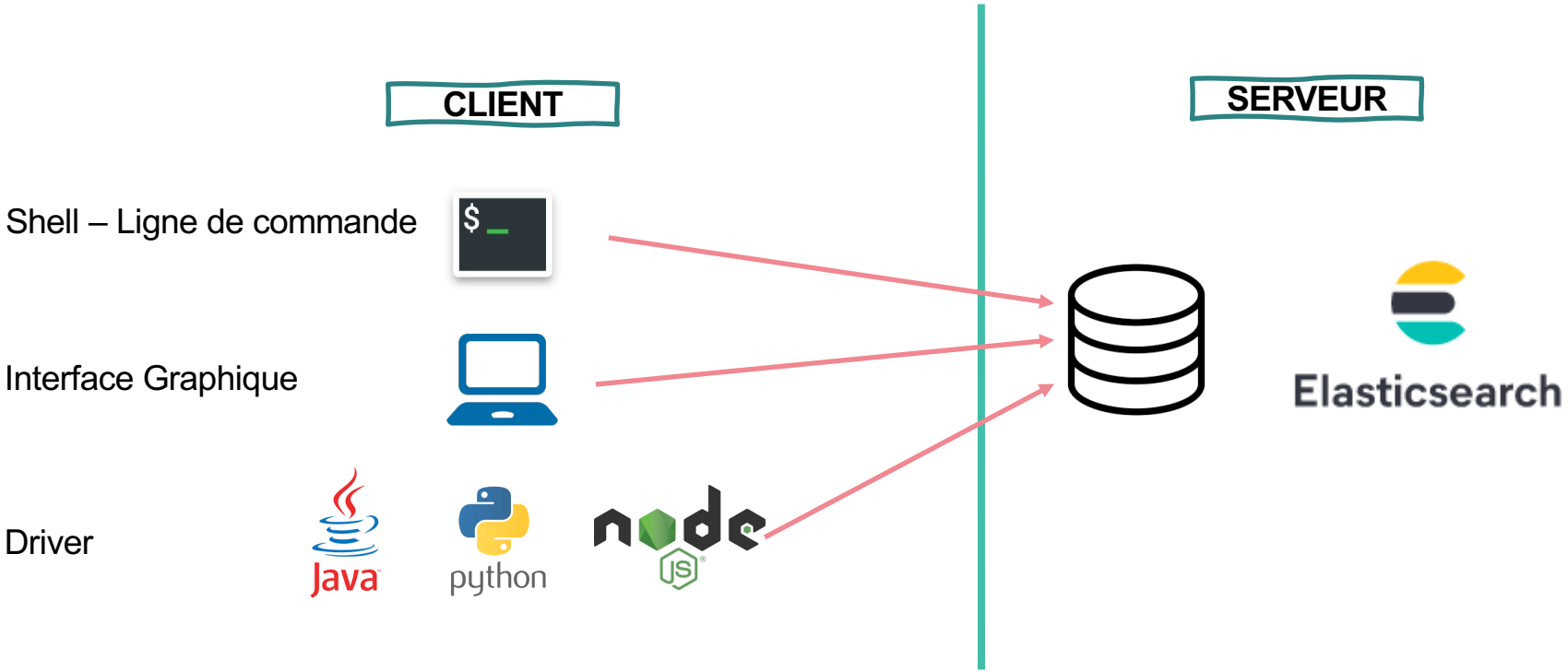
ElasticSearch

Version, maintenance et EOL

Elasticsearch	Date de fin du support technique		
	Date de début d'abonnement		Maintenue jusqu'à
	Avant le 1er octobre 2022	À partir du 1er octobre 2022	
7.13.x	25/11/2022	10/02/2023	7.14.0
7.14.x	03/02/2023	10/02/2023	7.15.0
7.15.x	2023-03-22	10/02/2023	7.16.0
7.16.x	07/06/2023	10/02/2023	7.17.0
7.17.x	01/08/2023	10/02/2023	9.0.0

ElasticSearch

Client/Serveur



ElasticSearch

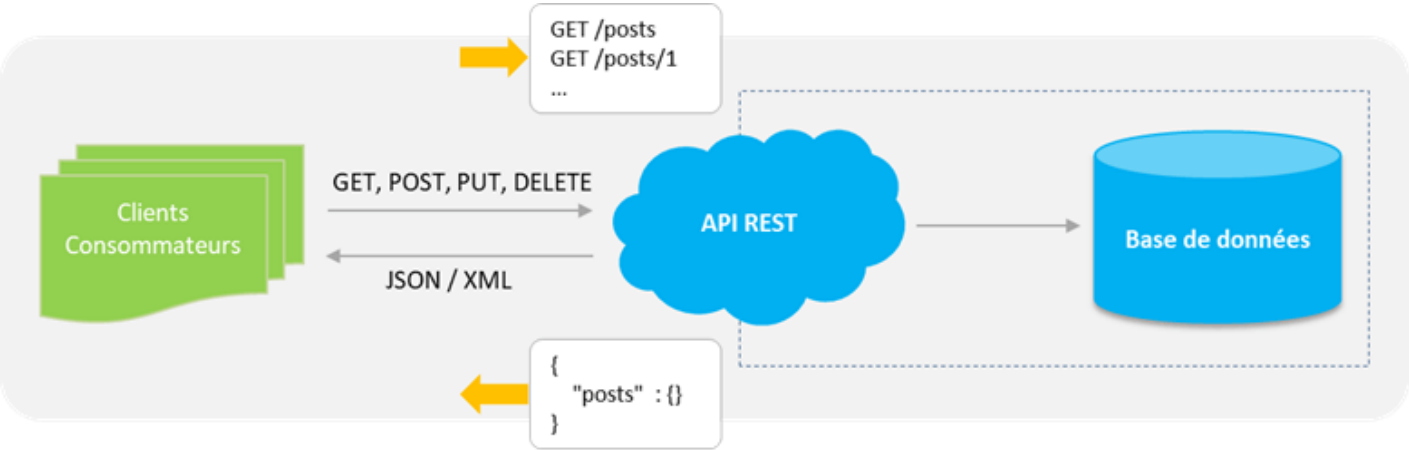
Lucene

- Librairie JAVA publiée pour la première fois en mars 2000
- Permet d'indexer et rechercher du texte
- Fournit un ensemble d'utilitaire autour du texte :
  - Indexation FullText
  - Recherche exact
  - Recherche approchée
  - Gestion des accents
  - Mots se rapprochant
  - Base verbale
  - Stop Word
  - Mise en surbrillance de texte
  - Gestion des langues
  - …
- Il est au cœur de nombreux moteurs de recherche et sites internet.

ElasticSearch

RestFul

- Correspond à des API REST : Communication client/serveur.
- Les API REST sont basées sur HTTP
- Un client lance une requête HTTP et le serveur renvoit une réponse.
- Le format de réponse est généralement du JSON



ElasticSearch

RestFul – Les verbes

- Les requêtes reposent principalement sur 4 verbes (Les opérations CRUD) :
  - GET : Récupération de données. Les données sont dans l'URL (Path Parameter)
  - POST : Création de données. Le corps (body) de la requête contient les données
  - PUT : Modification de données. Le corps (body) de la requête contient les données
  - DELETE : Suppression de données. Les données sont dans l'URL(Path Parameter)

- Exemple :
  - Récupérer le livre 4 : GET **http://bibliotheque/livres/4**
  - Créer le livre 10 : POST **http://bibliotheque/livres/10**
  - supprimer le livre 5 : DELETE **http://bibliotheque/livres/5**

```
curl -X POST \
-d '{"userName":"John DOE","comment":« Bla … Bla …"}' \
"http://localhost:8080/my-api/books/12/comments"
```

ElasticSearch

RestFul – Les URLS

- Les URLS se découpent logiquement.
- On utilise des mots, pas de verbes
- Un mot au pluriel désigne une ressource : books, cars, ….

- Exemple :
  - Tous les livres : *http://mywebsite.com/books*
  - Un livre : *http://mywebsite.com/books/87*
  - Tous les commentaires d'un livre : *http://mywebsite.com/books/87/comments*
  - Un commentaire pour un livre : *http://mywebsite.com/books/87/comments/1568*

- Ne pas faire :
  - *http://mywebsite.com/getAllBooks*

ElasticSearch

RestFul – Les codes retours HTTP

Code de retour	Description et utilisation
200 OK	Le serveur a traité la requête avec succès.
201 CREATED	Une nouvelle ressource a été créée.
204 No Content	Peut être utilisé en réponse à une requête DELETE effectuée avec succès.
206 Partial Content	En réponse à une requête demandant une réponse trop lourde pour être envoyée en une seule fois. De la pagination va être nécessaire pour récupérer l'ensemble des informations
304 Not Modified	Le client peut utiliser les données en cache car elles n'ont pas été modifiées depuis la date spécifiée.
400 Bad Request	La requête est invalide et ne peut pas être traitée par le serveur.
401 Unauthorized	La requête nécessite que le client soit identifié.
403 Forbidden	Le serveur a compris la requête mais l'utilisateur n'est pas autorisé à accéder à cette API.
404 Not Found	La ressource demandée n'existe pas.
500 Internal Server Error	Votre code ne devrait jamais renvoyer cette erreur. Cette erreur devrait être récupérée par votre code et traitée, pour ensuite renvoyer une réponse adéquate au client.

TD : REST

TD : REST

TD 1

- TD1.1 : Compléter le tableau

URLs	Actions
GET http://mysite.fr/room-api/rooms	
	Créer une salle
POST http://mysite.fr/room-api/rooms/12/events	
	Lister tous les événements de la salle numéro 15
	Modifier l'événement 5 de la salle 12
DELETE http://mysite.fr/room-api/rooms/12/events/18	

TD : REST

TD 1 : Correction

- TD1.1 : Compléter le tableau

URLs	Actions
GET http://mysite.fr/room-api/rooms	Lister toutes les salles
POST http://mysite.fr/room-api/rooms	Créer une salle
POST http://mysite.fr/room-api/rooms/12/events	Créer un événement pour la salle 12
GET http://mysite.fr/room-api/rooms	Lister tous les événements de la salle numéro 15
PUT http://mysite.fr/room-api/rooms/12/events/5	Modifier l'événements 5 de la salle 12
DELETE http://mysite.fr/room-api/rooms/12/events/18	Supprimer l'événement 18 de la salle 12

Organisation de la donnée

Organisation de la donnée

Vocabulaire

Organisation logique de la donnée

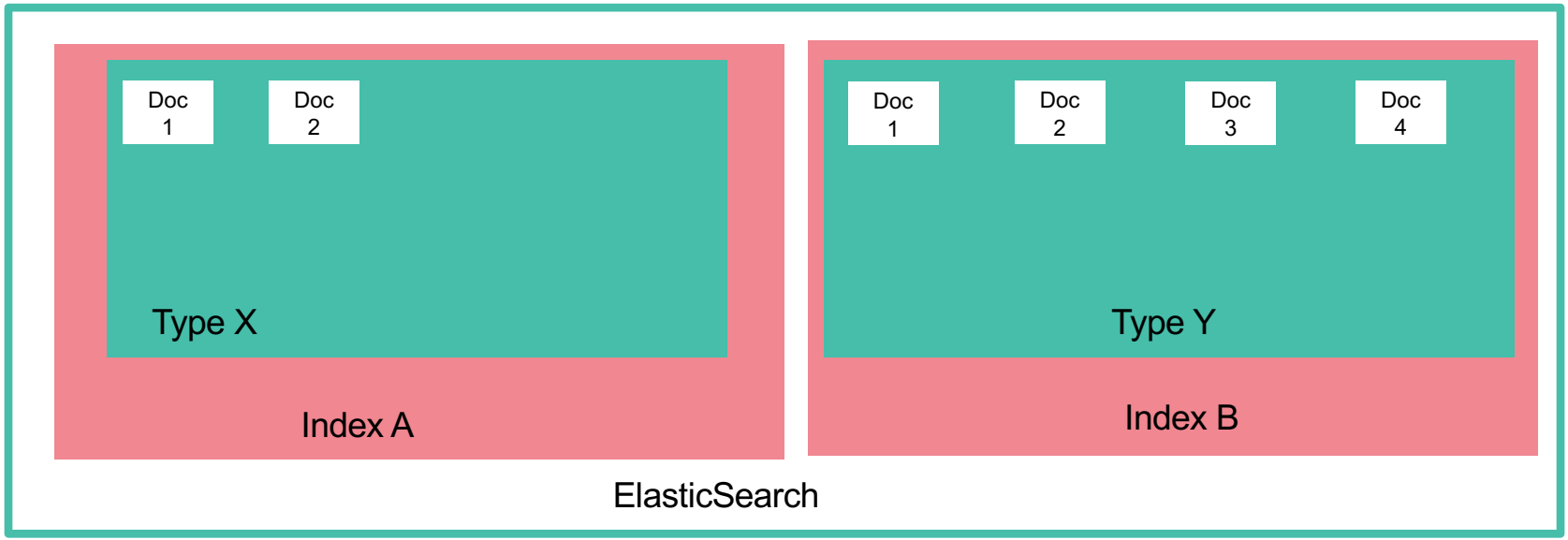
- **L'index** : Dans la terminologie ElasticSearch, un index est un ensemble de documents JSON.
- **Le type** : encore appelé Mapping, fait référence à un ensemble de documents qui partagent un ensemble de champs commun, présents dans le même index.
- **Un index contient un seul type**
- **Le document** : C'est un ensemble de champs ou propriétés définis de façon spécifique dans le format JSON. Chaque document appartient à un type et réside dans un index. A chaque document est associé un identifiant unique appelé UID ;

Organisation physique de la données

- **Le noeud** : en ElasticSearch, un noeud fait référence à une instance d'exécution du logiciel ElasticSearch.
- **Le cluster** : un cluster ElasticSearch est un ensemble d'un ou plusieurs nœuds.
- **La partition (Shard)** : une partition est une partie de l'index.
- **La réplique** : les indexes et les partitions sont répliqués à travers les nœuds du cluster pour accroître la haute disponibilité des traitements en cas de panne. De plus, les répliques permettent de paralléliser les traitements de recherche de contenu dans le cluster, ce qui accroît la performance d'ElasticSearch ;

Organisation de la donnée

Organisation logique des documents



Organisation de la donnée

Exemple d'organisation

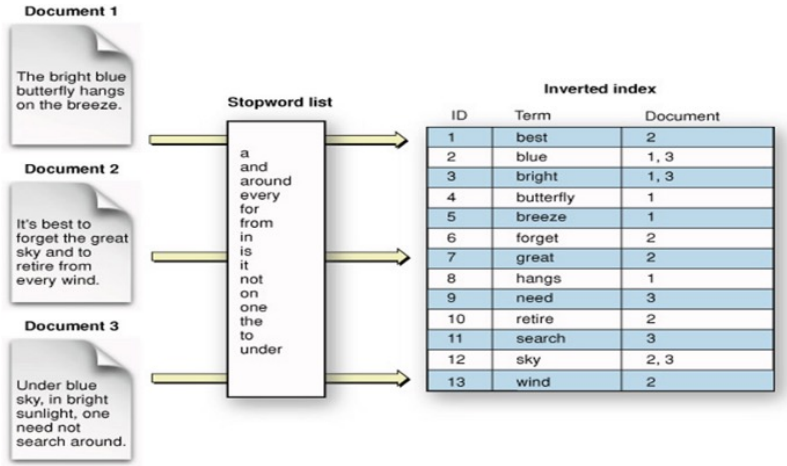
- Une application avec différents types métier
  - Exemple : Une application qui stocke des personnes, des offres d'emploi , des entreprises
  - Utilisation d'une répartition de la donnée par type
  - On aura un index pour chaque type : personnes, offres, entreprises

- Une application qui stocke des données orientées time series
  - Exemple : Des mesures de température dans les villes de France
  - Utilisation d'une répartition de la donnée par date
  - Toutes les données sont de même type, mais on souhaite les répartir pour des questions de volumétries et de performance
  - On peut avoir un index par mois et un type/indice par jour avec les mesures de la journée
  - Permet de limiter le volume de documents lorsque l'on recherche par date.

Organisation de la donnée

Index inversé (Lucene)

- Elasticsearch permet une recherche rapide car au lieu de chercher directement un mot dans un texte, il cherche ce mot dans un index. Par analogie avec la recherche d'un mot dans un livre, la recherche Elasticsearch consisterait à relever les numéros des pages concernées dans un index des mots-clé, placé en fin de livre, plutôt que de parcourir tous les mots de toutes les pages.
- Ce type d'index des mots-clés est appelé **index inversé** : la structure de donnée est centrée sur les mots-clé plutôt que sur les pages.



Organisation de la donnée

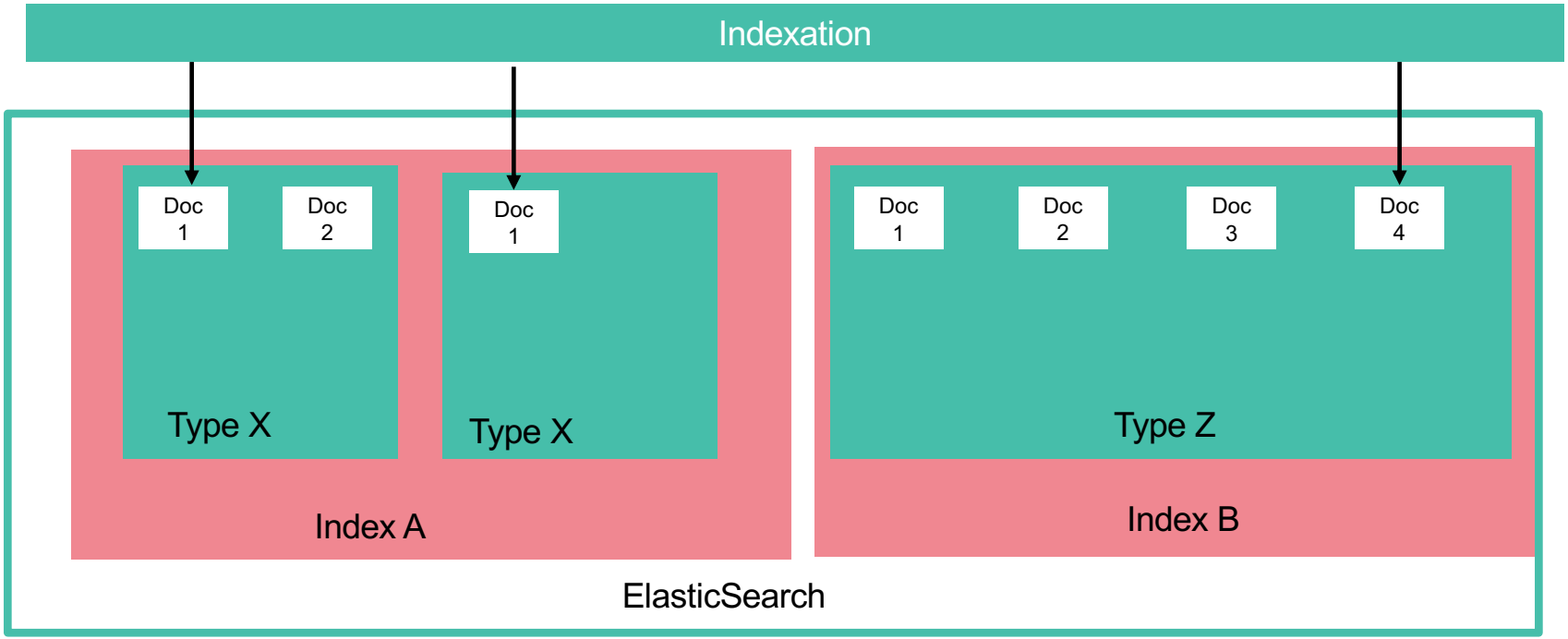
Indexation

- Lorsqu'on envoie un document au serveur pour indexation, le serveur va analyser le premier document pour en créer un schéma par défaut suivant les propriétés du document.
- Les documents suivants doivent correspondre au schéma.
- Trois champs supplémentaires sont utilisés pour identifier le document de manière unique:
  - un *index :* une grande catégorie dont fait partie le document (exemple: articles)
  - un *type* : une sous-catégorie (exemple: articles de sport) = correspond à l'indice
  - un *id* : un identifiant (exemple: 1245 — peut être généré automatiquement par le serveur)

- Le schéma créé après le premier import, ne peux pas être modifié. Il doit être supprimé puis recréé. L'ensemble des documents doit donc être réimporté
- L'ajout à l'index se fait périodiquement (quelques millisecondes) : Attention un document indexé n'est pas forcement accessible immédiatement.

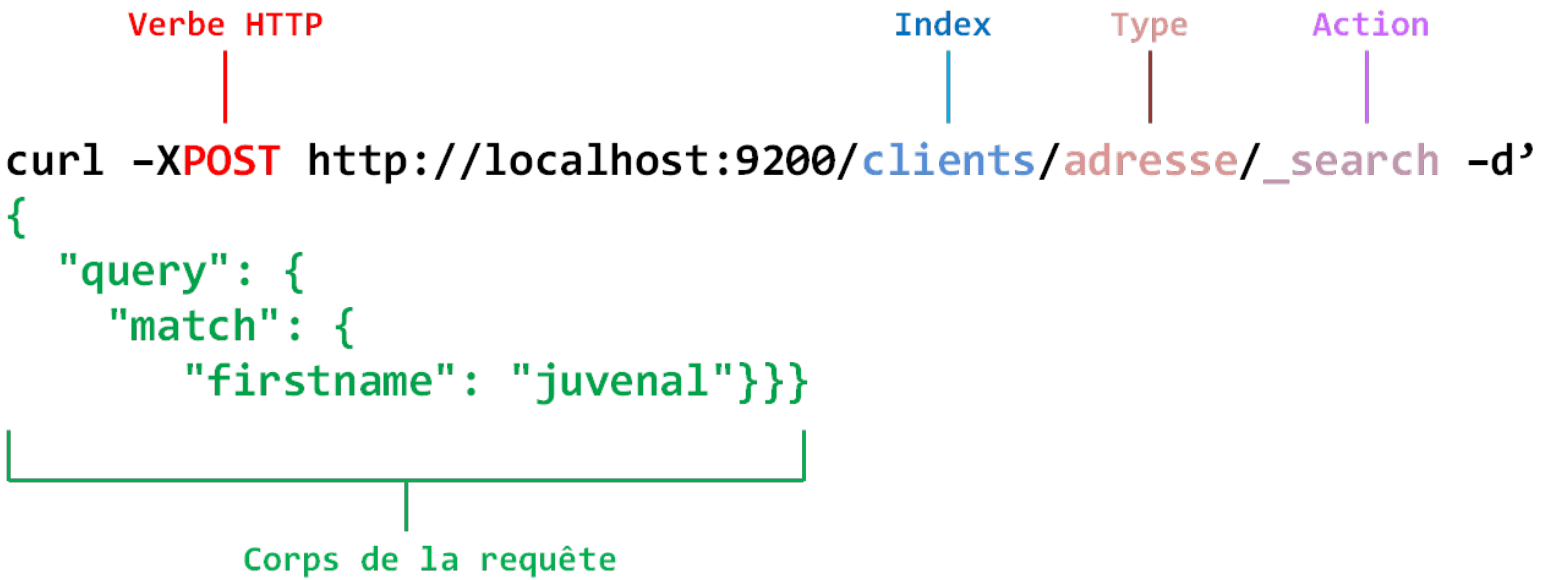
Organisation de la donnée

Indexation



Organisation de la donnée

Rest



Organisation de la donnée

Importer un document avec un identifiant

- On ajout un document avec pour identifiant « 1 »

```
curl -XPUT "http://localhost:9200/megacorp/employee/1?pretty"
-d' { "first_name" : "John", "last_name" : "Smith", "age" : 25, "about" : "I love to go rock climbing", "interests":
[ "sports", "music" ] }'
```

- Réponse du serveur


```
{
  "_index": "megacorp",
  "_type": "employee",
  "_id": "1",
  "_version": 1,
  "result": "created",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "_seq_no": 0,
  "_primary_term": 1
}
```

Organisation de la donnée

Importer un document sans identifiant

- On ajout un document sans identifiant

```
curl -XPUT "http://localhost:9200/megacorp/employee?pretty"
-d'{ "first_name" : "Eric", "last_name" : "Dupond", "age" : 30, "about" : "I love Game ansd Ski", "interests": [ "games", "ski" ] }'
```

- La réponse contient l'identifiant unique généré par le serveur


```
{
  "_index": "megacorp",
  "_type": "employee",
  "_id": "W9TZGX0B0HAPZ726hMfF",
  "_version": 1,
  "result": "created",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "_seq_no": 1,
  "_primary_term": 1
}
```

Organisation de la donnée

Récupérer le Mapping

- permet de récupérer le mapping pour le type "employee » pour l'index « megacorp »

```
curl -XGET "http://localhost:9200/megacorp/employee/_mapping?pretty"
```

- La réponse retourne la mapping du type « employee » pour l'index « megacorp »


```
{
  "megacorp": {
    "aliases": {},
    "mappings": {
      "properties": {
        "about": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "age": {
          "type": "long"
        },
```

Organisation de la donnée

Mapping – Cas de l'ajout d'un nouveau champs

- On ajoute un nouveau champs dans le document : « naissance »

```
curl -XPUT "http://localhost:9200/megacorp/employee?pretty" -d'
{ "first_name" : "Eric", "last_name" : "Dupond", "age" : 30, "about" : "I love Game ansd Ski", "interests": [ "games", "ski" ],
”naissance”:”2OOO-10-21” }'
```

- Le mapping est modifié pour ajouter le nouveau champs :

```
....
   "naissance": {
    "type": "date"
 },
....
```

ElasticSearch

Mapping – Contrôle de cohérence

- On ajoute un document avec le champs « naissance » mais dans un format non conforme à une date

```
curl -XPUT "http://localhost:9200/megacorp/employee?pretty"
-
d'{ "first_name" : "Eric", "last_name" : "Dupond", "age" : 30, "about" : "I love Game ansd Ski", "interests": [ "games", "ski" ],”naissance”:”2
0 octobre 2000” }'
```

- Erreur en réponse

```
{
  "error": { "root_cause": [ {
          "type": "mapper_parsing_exception",
          "reason": "failed to parse field [naissance] of type [date] in document with id 'YNQVG30B0HAPZ72608f_'. Preview of field's value: '20 octobre 2000'"
        }
    ],
    ...
}
```

Mapping

Mapping

Dynamique ou Explicite

- Le mapping dynamique permet un typage de champs automatique par le serveur:
  - Pratique pour explorer un contenu
  - Pour tester les fonctionnalités
  - Mais limité dans le cadre d'une application

- Le mapping explicite est fourni par l'utilisateur pour définir l'indexation souhaitée suivant le contenu et l'utilisation du contenu.
  - Définie avec précision comment indexer les champs
  - Permet de limiter les index : Trop de champs indéxés va consommer de l'espace disque et de la mémoire
  - Pour de grands volumes de données, il faut savoir être raisonnable

Pour connaitre le mapping d'un index

```
GET /my-index-000001/_mapping?pretty
```

Pour connaitre le mapping d'un champ

```
GET /my-index-000001/_mapping/field/employee-id
```

Mapping

Dynamique

- La détection automatique des champs est définie suivant des règles paramétrables. Ci-dessous, les règles par défaut :

JSON data type	Mapping
null	Le champs n'est pas ajouté
true or false	Type boolean
double	float
integer	long
object	object
array	Il n'y a pas de notion de tableau. Dépend de la première valeur non null.
String correspondant à une date	date
String correspondant à une valeur numérique	float ou long
Autre String	text avec une propriété keyword

Mapping

Dynamique – date detection

- Une valeur String correspondant au format date ISO est détectée comme une date.
- Format par défaut : `"yyyy/MM/dd HH:mm:ss Z||yyyy/MM/dd Z”`

```
PUT my-index-000001/_doc/1 { "create_date": "2015/09/02" }
```

```
“create_date": {
  "type": "date"
},
```

- Il est possible de désactiver la détection en modifiant l’index

```
PUT my-index-000001 { "mappings": { "date_detection": false } }
```

- Ou de modifier les formats en modifiant l’index

```
PUT my-index-000001 { "mappings": { "dynamic_date_formats": ["MM/dd/yyyy"] } }
```

Mapping

Dynamique – Détection de valeur numérique

- Une valeur numérique fournie en format String peut être détectée automatiquement.
- Mais Désactivée par défaut, il faut donc l'activer :

```
PUT my-index-000001
{
 "mappings": {
  "numeric_detection": true
 }
}
```

- Ajouter des valeurs. On peut également ajouter indifféremment en format string ou numérique dans un même champ.

```
PUT my-index-000001/_doc/1
{
 "my_float_as_string":  "1.0",
 "my_integer_as_string": "1"
 "my_float":  1.0,
 "my_integer": 1
}
```

Mapping

Dynamique – Modification

- Un schéma peut être modifié : Ajout de nouveaux champs à un index existant
- Exemple : Ajout d'un champ employee-id de type keyword.

```
PUT /my-index-000001/_mapping
{
  "properties": {
    "employee-id": {
      "type": "keyword "
    }
  }
}
```

- Par contre
  - un champ existant ne peut pas être supprimé
  - Un type ne peut pas être modifié
- Ces changements nécessitent une recréation de l'index, du mapping et de réimporter les documents

Mapping

Dynamique – Exemple

- Création d'un mapping dynamique pour un document

Ajout d'un document

```
PUT my-index-000001/_doc/1
{
  "region": "US",
  "manager": {
    "age":     30,
    "name": {
      "first": "John",
      "last":  "Smith"
    }
  }
}
```

Mapping correspondant

```
GET my-index-000001
{
  "mappings": {
    "properties": {
      "region": {
        "type": "keyword"
      },
      "manager": {
        "properties": {
          "age":  { "type": "integer" },
          "name": {
            "properties": {
              "first": { "type": "text" },
              "last":  { "type": "text" }
            }
          }
        }
      }
    }
  }
}
```

Mapping

Explicite

- Il est possible de spécifier un mapping à la **création** d'un index
- Ajout d'un mapping pour l'index « my-index-000001 » avec 3 propriétés de type différent

```
PUT /my-index-000001
{
  "mappings": {
    "properties": {
      "age":    { "type": "integer" },
      "email":  { "type": "keyword"  },
      "name":   { "type": "text"  }
    }
  }
}
```

Mapping

L’indexation

- Chaque champ a un type qui indique la typologie des données contenues
- Le type permet d’influencer les recherches et les fonctionnalités disponibles

- Il est possible d’indexer un même champ avec différents types
  - Ex: on voudrait indexer un champ « text » comme un mot clé pour une recherche exacte et en indexation full text pour une recherche approchée.

- Il existe beaucoup de choix possibles pour indexer les champs.
- Le choix se fera suivant le type mais aussi suivant l’utilisation de la donnée

- Options possible:
  - Index : indique si le champ doit être indexé ou non
  - Alias : Ajoute un alias sur le nom du champ. Permet de renommer un champ sans recréer l’index.

Mapping

Explicite – Les types

- Des types simples et unitaires :

Type	
Binary	Valeur binaire encodée en base 64
Boolean	True ou False
Date	Une date suivant le format ISO par défaut

Mapping

Explicite – Les types

- La famille « Numeric » : Pour les nombres

Type	Description	Valeurs
long	Un entier signé de 64 bit	Min : -2^{63} / Max : $2^{63}-1$
integer	Un entier signé de 32 bit	Min : -2^{31} / Max : $2^{31}-1$
short	Un entier signé de 16 bit	Min : -32 768 / Max : 32 767
Byte	Un entier signé de 8 bit	Min : -128 / Max : 128
Double	A double-precision 64-bit IEEE 754 floating point number	Min : -1×10^{37} / Max : 1×10^{37}
float	A single-precision 32-bit IEEE 754 floating point number	Min : -1×10^{37} / Max : 1×10^{37}

Mapping

Explicite – Les types

- La famille « Objet » : Pour les objets JSON embarqués
- Il faut considérer les objets comme des sous parties du document

Ajout d'un document

```
PUT my-index-000001/_doc/1
{
 "region": "US",
 "manager": {
  "age":    30,
  "name": {
   "first": "John",
   "last":  "Smith"
  }
 }
}
```

Il faut considérer une représentation suivante

```
{
 "region":          "US",
 "manager.age":     30,
 "manager.name.first": "John",
 "manager.name.last":  "Smith"
}
```

Mapping correspondant

```
GET my-index-000001
{
 "mappings": {
  "properties": {
   "region": {
    "type": "keyword"
   },
   "manager": {
    "properties": {
     "age":  { "type": "integer" },
     "name": {
      "properties": {
       "first": { "type": "text" },
       "last":  { "type": "text" }
      }
     }
    }
   }
  }
}
```

Mapping

Explicite – Les types

- La famille « Keyword » : Pour des chaînes de caractères

Type	
keyword	Pour du texte structuré comme des tags, des adresse emails, …. Principalement pour des recherches sur des termes exacts
Constant_keyword	Pour des énumérations
Wildcard	Pour un contenu non structuré généré (log, requête HTTP, …)

Mapping

Explicite – Les types

- La famille « Text » : Pour les chaînes de caractères

Type	Description	Exemple
text	Le type classique pour une indexation Full-text.	Utilisé pour des contenu de mail, des descriptions, …
Completion	Pour de l'auto complétion dans le but de proposer une suggestion de terme à l'utilisateur	Suggère des mots présents dans un document
token	Indexe le nombre de mots dans le texte	Compter le nombre de mot dans un résumé

Mapping

Explicite – Les types

- La famille « Spatial » : Les données de géolocalisation

Type	Description	Exemple
geo_point	Utilise une paire de valeur latitude/longitude	Géolocalisation d'un magasin par sa latitude/longitude
point	Pour des points cartésien X/Y	
geo_shape	Forme géométrique (ligne, polygone, …) suivant une latitude / longitude	Emplacement d'un terrain
shape	Forme géométrique (ligne, polygone, …) des points cartésiens X/Y	

Mapping

Explicite – Les types

- Ajouter un document avec un geo_point

```
PUT my-index-000001/_doc/1
{
  "text": "Geopoint as an object",
  "location": {
    "lat": 41.12,
    "lon": -71.34
  }
}
```

```
PUT my-index-000001/_doc/2
{
  "text": "Geopoint as a string",
  "location": "41.12,-71.34"
}
```

```
PUT my-index-000001/_doc/4
{
  "text": "Geopoint as an array",
  "location": [ -71.34, 41.12 ]
}
```

- Ajouter un document avec un geo_point

```
POST /example/_doc
{
  "location" : {
    "type" : "linestring",
    "coordinates" : [
      [-77.03653, 38.897676], [-77.009051, 38.889939]]
  }
}
```

```
POST /example/_doc
{
  "location" : {
    "type" : "polygon",
    "coordinates" : [
      [ [-77.03653, 38.897676], [-77.03653, 37.897676],
        [-76.03653, 38.897676], [-77.03653, 38.997676],
        [-77.03653, 38.897676] ]
    ]
  }
}
```

Mapping

Percolator

- Il s’agit d’une recherche inversée.
- On n’indexe plus des documents mais des requêtes dans un index
- A chaque ajout d’un document, il est possible d’interroger l’index « percolator »
- La réponse retourne les requêtes qui correspondent au document
- Utilisation métier : alerte sur des recherches sauvegardées
  - Être alerté lorsqu’un produit correspond à ma recherche (le bon coin)

Analyse de texte

Analyse de texte

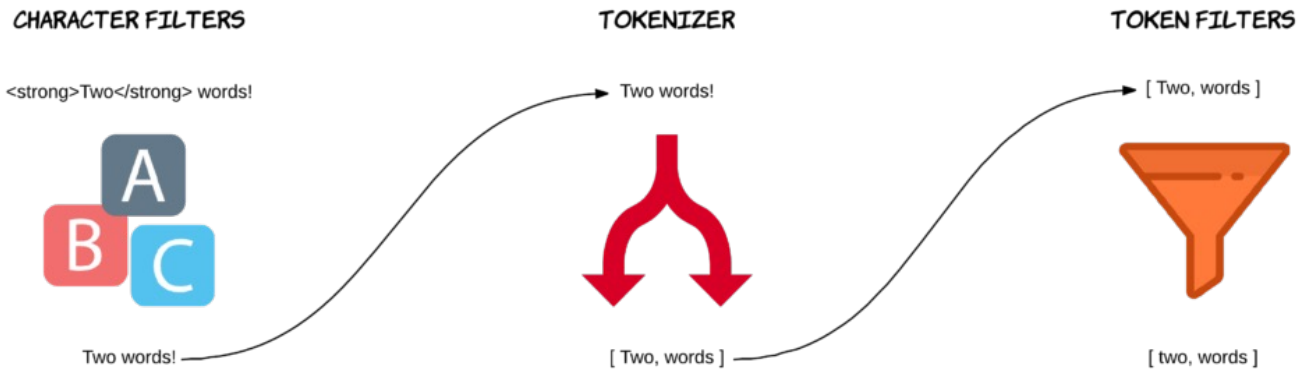
Un analyser

- L’analyser consiste en une suite de 3 traitements.
- Il existe des analyser par défaut, mais ils peuvent être paramétrés
- Ils manipulent du texte, ils sont donc dépendants de la langue du texte : Français, Anglais, …

- Les 3 étapes :
  - Characters Filters : Reçoit un flux avec le texte original et le transforme en ajoutant, supprimant, modifiant les caractères (Suppression de marqueur HTML).
  - Tokenizer : Reçoit le flux précédent et sépare les caractères en mots suivant les espaces et la ponctuation. Le tokenizer conserve également l’ordre des mots et leurs emplacements.
  - Token Filter : Reçoit le flux de mots précédent et il ajoute, supprime ou modifie les mots (Convertions en majuscule/minuscule, suppression de mot insignifiant, synonyme, base verbale, …)

Analyse de texte

Un analyser - exemple



Analyse de texte

Characters Filter

- Permet de transformer un texte de caractères ou de textes sans valeurs : Ajout, suppression, modification
- Suppression :
  - Page HTML : Suppression des balise HTML : Titre
- Modification :
  - Remplacement de code par une valeur
  - Pour des couleurs #FF0000 remplacé par le mot rouge
- Ajout :
  - Ajout des termes correspondant à des acronymes
  - RAS = Rien à signaler
- Il existe des characters filter prés défini par ElasticSearch

Analyse de texte

Tokenization

- La tokenization rend possible la recherche full text en isolant chaque mot d'un texte. On parle de token.
- Prenons la phrase suivante : Les chevaux marrons mangent de l'herbe
- Si on une recherche exact avec « chevaux mangent », on ne retrouvera pas le document
- La tokenisation de la phrase : [Les, chevaux, marrons, mangent, de, l'herbe]
- La tokenization de la requête [chevaux, mangent]
- Maintenant la recherche par termes permettra de retrouver le document

Analyse de texte

Token Filter

- La tokenization permet la recherche sur des termes individuels, mais chaque tokens doit correspondre exactement.
- La tokenisation de la phrase : [Les, chevaux, marrons, mangent, de, l'herbe]

- Si je cherche Chevaux : pas de correspondance
- Si je cherche mange : pas de correspondance
- Si je cherche broutent : pas de correspondance

- Pour améliorer la recherche et résoudre ces problèmes, les tokens sont normalisés à l'étape du « token filter » :
  - Remplacement des majuscules par des minuscule
  - Remplacement des accents
  - Ajout des racines de mots et base verbale : Stemming
  - Ajout de synonymes
  - Suppression de mot inutiles : stop words :
    - Anglais : a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with
    - Français : Lors, au, aucuns, aussi, autre, avant, avec, avoir, bon, car, ce, cela, ces, …

  Le token filter : [cheval, chevaux, marrons, marron, mangent, mange, manger, l'herbe]

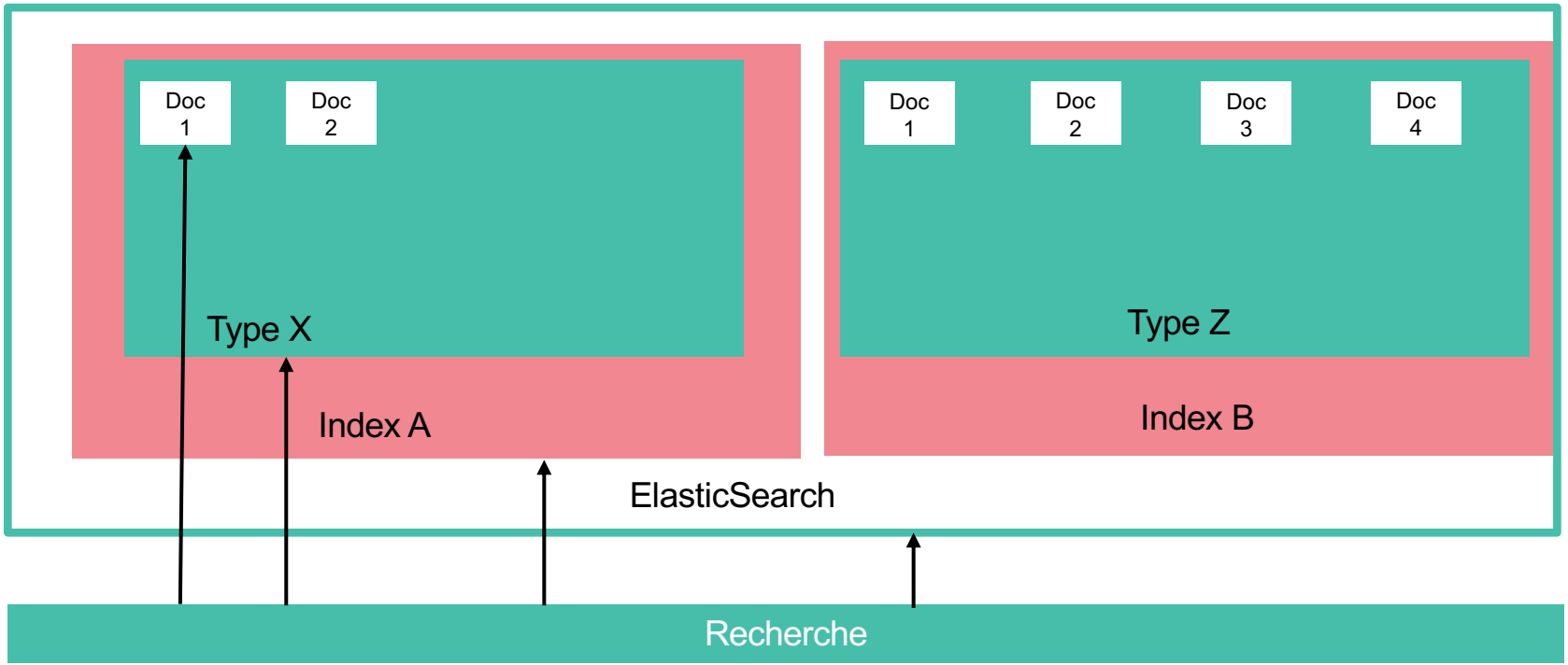
Recherche de données

Recherche de données

RestFul - Exemple

Recherche de données

Les cibles de recherche



Recherche de données

Accéder à un document

- Accéder directement à un document suivant son identifiant :

```
curl -XGET "http://localhost:9200/megacorp/employee/1?pretty"
```

- La réponse retourne un document JSON reprenant les informations suivantes :
  - _index : L'index d'appartenance du document
  - _type : Le type de document
  - _id : L'identifiant unique du document
  - _version : Indique la version du document. Incrémenté lorsqu'un document est remplacé
  - _source : l'intégralité du document d'origine indexé par ElasticSerarch

```
{
 "_index": "megacorp",
 "_type": "employee",
 "_id": "1",
 "_version": 1,
 "found": true,
 "_source": {
  "first_name": "John",
  "last_name": "Smith",
  "age": 25,
  "about": "I love to go rock climbing",
  "interests": ["sports","music"]
 }
}
```

Recherche de données

Lister les documents

```
curl -XGET "http://localhost:9200/megacorp/employee/_search?pretty"
```

- Lorsque l'on utilise l'action de recherche, le résultat de la requête se décompose en trois parties

Première partie technique

- On y retrouve les informations suivantes :
  - Took : Le temps en milliseconde pour réaliser la recherche
  - Time_out : Si la recherche a terminé en timeout. Elle peut cependant retourner des résultats
  - _ Shards : Indique les shards intérrogés

```
{
"took": 2,
"timed_out": false,
"_shards": {
  "total": 5,
  "successful": 5,
  "failed": 0
  }
...
```

Recherche de données

Lister les documents

```
curl -XGET "http://localhost:9200/megacorp/employee/_search?pretty"
```

Deuxième partie

- Des informations sur le résultat de la recherche :
  - Hits : un objet JSON contenant le résultat de la recherche
  - Total : le nombre de documents retournés
  - Max_score : Le document avec le meilleur score qui correspond à la recherche
  - Hits : le tableau de résultat.

```
"hits": {
"total": 2,
"max_score": 1,
"hits "  : [ .....]
}
```

Recherche de données

Lister les documents

```
curl -XGET "http://localhost:9200/megacorp/employee/_search?pretty"
```

Troisième partie

- Un tableau avec les résultats et pour chaque résultat, les mêmes informations qu'un document unitaire :
  - _index : L'index d'appartenance du document
  - _type : Le type de document
  - _id : L'identifiant unique du document
  - _source : l'intégralité du document d'origine indexé par ElasticSerarch

- Et en plus un champ score :
  - Correspond à un score de pertinence du document par rapport à la recherche.
  - Plusieurs critères : nombre d'occurrence du mot, proximité du mot, …
  - Les documents sont triés par défaut par score en ordre décroissant
  - La valeur se situe entre 0 et 2 (ex: 0,625)


```
{
 "_index": "megacorp",
 "_type": "employee",
 "_id": "1",
 "_score": 1,
 "_source": {
  "first_name": "John",
  "last_name": "Smith",
  "age": 25,
  "about": "I love to go rock climbing",
  "interests": ["sports","music"]
 }
}
```

Recherche de données

Rechercher des documents suivant une propriété

- Rechercher tous les documents dont le champs « last_name » contient/est égale à la valeur « smith »

```
curl -XGET "http://localhost:9200/megacorp/employee/_search" -d'
{
  "query" : {
    "match" : {
      "last_name" : "Smith"
    }
  }
}
```

- Attention : la recherche est une combinaison entre le mapping du champ et entre les opérateurs de la requête
- L'opérateur « match » n'est pas interprété de la même façon suivant le type du champ

Recherche de données

Trier les résultats

- Par défaut, les résultats sont triés par le score en ordre décroissant : La pertinence
- On peut ajouter un ou plusieurs critères de tri qui peuvent être ascendants (« asc ») ou descendants (« desc »)
- L'ordre de tri sur des champs est par défaut ascendant
- Il est possible de trier par ordre d'insertion avec le mot clé _doc
- Le tri par score utilise le mot clé _score

- Exemple :
  - Je recherche les documents correspondants à l'utilisateur « kimchy »
  - Je trie par le champ user (ascendant par défaut)
  - Ensuite par le nom et l'âge en descendant
  - Puis par le « score » (descendant par défaut)

```
GET /my-index-000001/_search
{
 "query" : {
  "match" : { "user" : "kimchy" }
 },
 "sort" : [
  "user",
  { "name" : "desc" },
  { "age" : "desc" },
  "_score"
 ]
}
```

Recherche de données

Filtrer les champs retournés

- Par défaut, le résultat contient toutes les données source de chaque document. (champ _source)
- Problématique : Le résultat de la recherche peut être volumineux
  - Ex : le document contient un résumé détaillé du film

- Il est possible de filtrer les champs retournés en ajoutant en précisant le champ _source après la requête

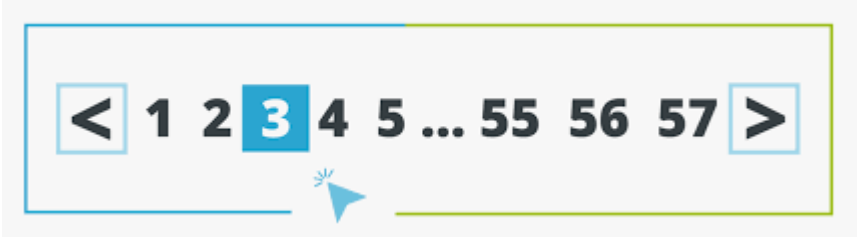
- Exemple : Je recherche tous les documents dans l'index bank mais je ne récupère que les champs « account_number » et « balance » des documents d'origine

```
curl -XPOST 'localhost:9200/bank/_search?pretty' -d '
{
  "query": { "match_all": {} },
  "_source": ["account_number", "balance"]
}'
```

Recherche de données

La pagination

- Par défaut, la recherche retourne les 10 meilleurs résultats
- Il est possible de parcourir les documents suivants avec deux paramètres :
  - From : le nombre d'éléments à ignorer
  - Size : la taille d'une page
- Utiliser pour les paginations de résultat sur les sites de recherche



- Exemple : Je veux récupérer les 20 premiers documents à partir du sixième dont le user.id est « kimchy »

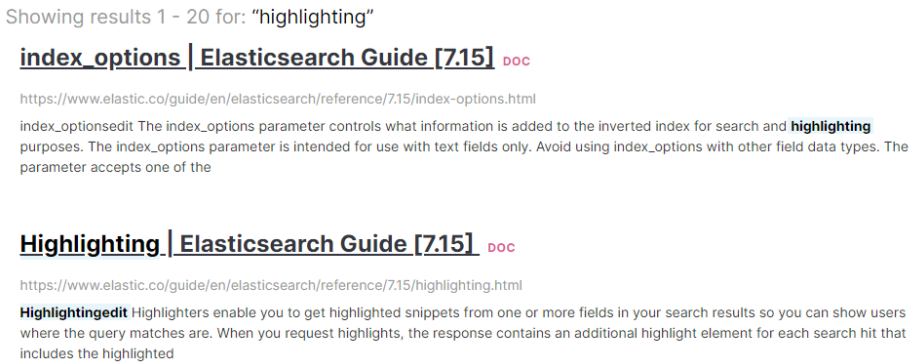
```
GET /_search
{
  "from": 5,
  "size": 20,
  "query": {
    "match": {
      "user.id": "kimchy"
    }
  }
}
```

Recherche de données

Le Highlighting

- Permet de mettre en surbrillance dans le résultat une partie du texte contenant les mots recherchés
- Mise en évidence à l'utilisateur du contenu qui l'intéresse
- Le serveur ajoute des tags dans le texte pour encadrer la portion de texte en surbrillance.
- Par défaut : et , mais on peut les changer

```
GET /_search
{
  "query" : {
    "match": { "body": "highlighting" }
  },
  "highlight" : {
    "pre_tags" : ["<tag1>"],
    "post_tags" : ["</tag1>"],
    "fields" : {
      "body" : {}
    }
  }
}
```



Recherche de données

Recherche full text

```
curl -XGET 'localhost:9200/text/article/_search?pretty' -d '
{
  "query": {
    "match": {
      "random_text": "him departure"
    }
  }
}'
```


```
"hits": [{
    "_index": "text",
    "_type": "article",
    "_id": "4",
    "_score": 1.4513469,
    "_source": {
      "title": "Old education",
    "random_text": "Old education him departure any arranging one prevailed."
    }
},{
    "_index": "text",
    "_type": "article",
    "_id": "3",
    "_score": 0.28582606,
    "_source": {
      "title": "Repulsive questions",
      "random_text": "Repulsive questions contented him few extensive supported."
    }
}]
```

Recherche de données

Query DSL - match

Le plus courant et fréquemment utilisé :

- Sur un champ de type « text », il exécute une recherche Full text
- Sur un champ de type valeur exact (« keyword »), il exécute une recherche exacte

requête	Recherche	
{ "match": { “titre": “House of cards" }}	Text	Contient « House » ou « cards »
{ "match": { "date": "2014-09-01" }}	Exact	La date est égale à …
{ "match": { "visible": true }}	Exact	La valeur boolean est à true

Recherche de données

Query DSL – Term et Terms

Rechercher sur une valeur exacte avec une ou plusieurs valeurs
Par défaut, l'opérateur « or » est utilisé entre les termes recherchés.

requête	Description
{ "term": { "tag": "math" }}	Recherche tous les documents dont le champs « tag » est égale à « math ».
{ "terms": { "tag": ["math", "statistics"] }}	Recherche tous les documents dont le champs « tag » est égale à « math » ou « statistics ».

Recherche de données

Query DSL – MultiMatch

Recherche un même texte sur plusieurs champs

- Exemple : Je recherche le texte « probability theory » sur les champs title et body

```
{
  "multi_match": {
    "query": "probability theory",
    "fields": ["title", "body"]
  }
}
```

Recherche de données

Query DSL – Bool query Clause

Recherche combinée sur plusieurs champs avec 3 termes :

- must : Correspond à un « AND»
- must_not :Correspond à un « NOT »
- should : Correspond à un « OR »

```
{
 "bool": {
  "must": [
   { "term": { "tag":   "math" }},
   { "term": { "level": "beginner" }}
  ]
  "must_not": { "term": { "tag":   "probability"  }},
  "should": [
   { "term": { "favorite": true   }},
   { "term": { "unread":  true   }}
  ]
 }
}
```

Recherche de données

Recherche géospatiale

- Recherche géospatiale suivant des valeurs indexées dans le document.
- Utilisé pour des calculs de distance par exemple : Je veux toutes les offres d'emploi dans un rayon de 30 km.

Query	
geo_bounding_box	Les documents inclus dans un rectangle
geo_distance	Les documents inclus dans un cercle
geo_polygon	Les documents inclus dans un polygone
geo_shape	Intersection de formes géométriques

Agrégation

Agrégation

Présentation

Après avoir recherché, filtré et trié les documents, il est possible de les agréger pour présenter le résultat.

Il existe 3 types d'agrégation :

- Metric : permet de calculer des métriques sur les documents (somme, moyenne, ..)
- Bucket : Regroupe les documents par catégories
- Pipeline : Permet d'enchainer des agrégations.

Agrégation

Metric

- Agrégation de valeur numériques pour créer une nouvelle valeur numérique
- Deux types d'agrégation :
  - Le résultat est une valeur numérique simple : Min, Max, Sum, Avg, …
  - Le résultat est un ensemble de valeurs numériques : Stats

- Exemple : agréger tous les documents et exécuter l'opérateur « stats ».

```
POST /exams/_search
{
  "aggs": {
    “values_stats”: { "stats": { "field": “value" } }
  }
}
```

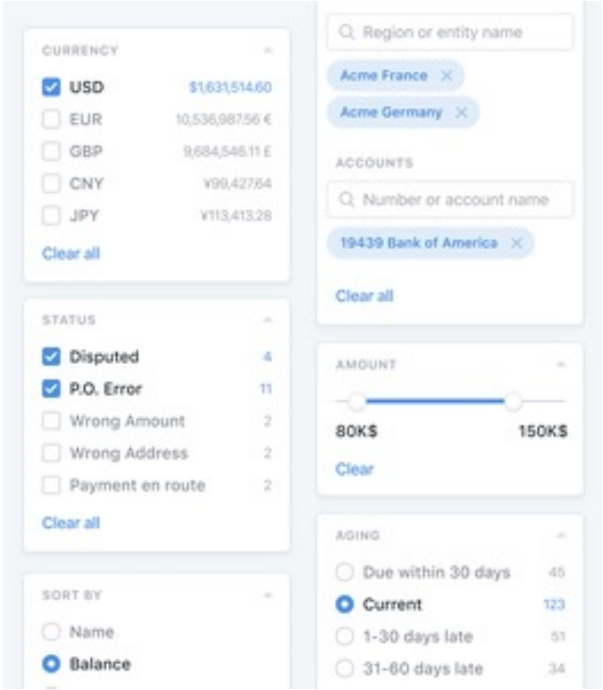
```
{
  ...
  "aggregations": {
    “values_stats”: {
      "count": 2,
      "min": 50.0,
      "max": 100.0,
      "avg": 75.0,
      "sum": 150.0
    }
  }
}
```

- Metrics aggregations
  - **Avg**
  - Boxplot
  - Cardinality
  - Extended stats
  - Geo-bounds
  - Geo-centroid
  - Geo-Line
  - Matrix stats
  - Max
  - Median absolute deviation
  - Min
  - Percentile ranks
  - Percentiles
  - Rate
  - Scripted metric
  - Stats
  - String stats
  - Sum
  - T-test
  - Top hits
  - Top metrics
  - Value count
  - Weighted avg

Agrégation

Bucket

- L’agrégation bucket permet de catégoriser les documents.
- On parle de facette.
- L’objectif est de regrouper tous les documents par critère et de compter les occurrences
- Couramment utilisé dans les sites de e-commerce.
- Le but est d’utiliser ensuite les facettes pour restreindre la recherche
- Exemple de regroupement :
  - Valeur exacte : Catégorie « Télévision »
  - Intervalle : Catégorisé par tranche de prix
  - Catégorisé par année



Agrégation

Bucket – exemple

Je veux agréger les articles en suivant le genre d'article

```
GET /_search
{
  "aggs": {
    "genres": {
      "terms": { "field": "genre" }
    }
  }
}
```

```
{
  ...
  "aggregations": {
    "genres": {
      "doc_count_error_upper_bound": 0,
      "sum_other_doc_count": 0,
      "buckets": [
        {
          "key": "electronic",
          "doc_count": 6
        },
        {
          "key": "rock",
          "doc_count": 3
        },
        {
          "key": "jazz",
          "doc_count": 2
        }
      ]
    }
  }
}
```

Agrégation

Bucket - exemple

Je veux agréger les articles par tranche de prix

```
GET sales/_search
{
  "aggs": {
    "price_ranges": {
      "range": {
        "field": "price",
        "ranges": [
          { "to": 100.0 },
          { "from": 100.0, "to": 200.0 },
          { "from": 200.0 }
        ]
      }
    }
  }
}
```

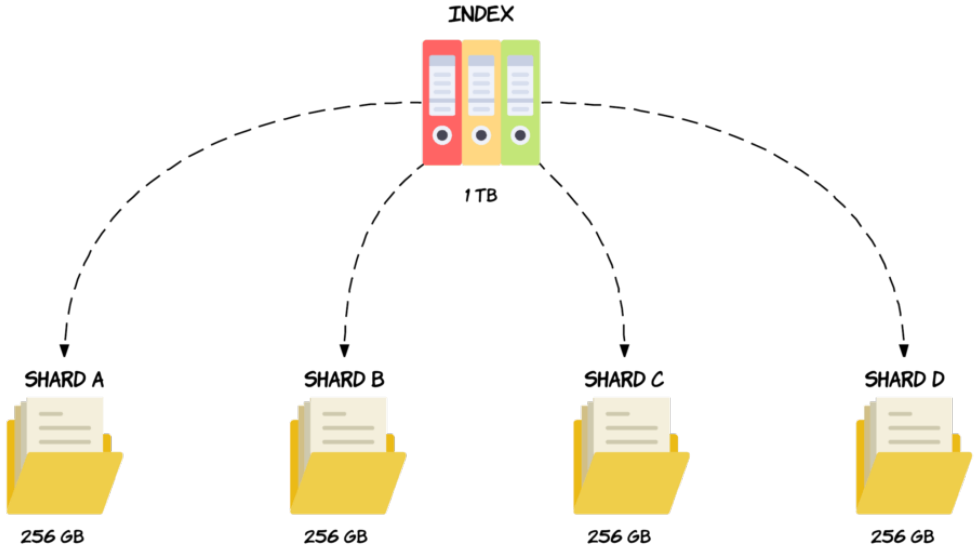
```
{
  ...
  "aggregations": {
    "price_ranges": {
      "buckets": [
        {
          "key": "*-100.0",
          "to": 100.0,
          "doc_count": 2
        },
        {
          "key": "100.0-200.0",
          "from": 100.0,
          "to": 200.0,
          "doc_count": 2
        },
        {
          "key": "200.0-*",
          "from": 200.0,
          "doc_count": 3
        }
      ]
    }
  }
}
```

Sharding et replication

Sharding et replication

Sharding

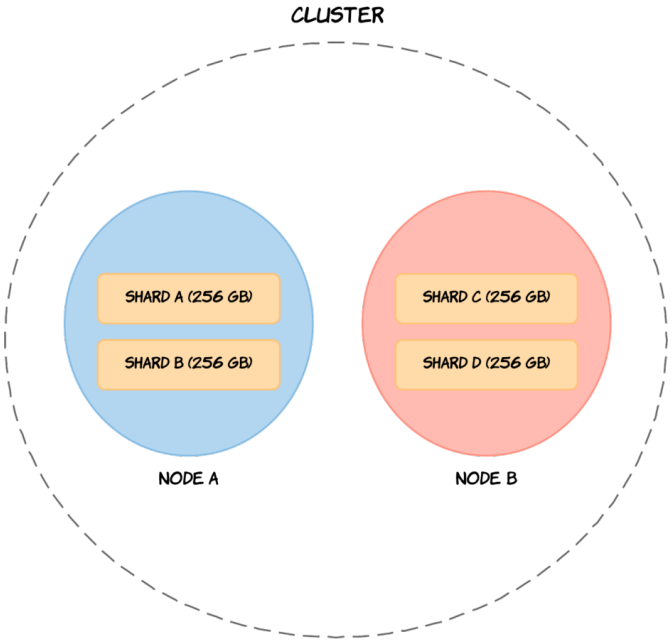
- Par défaut, ElasticSearch effectue du sharding sur les données, même sur un nœud unique.
- Par défaut, un index est divisé en 5 shards



Sharding et replication

Sharding

- Si on ajoute un second nœud, on parle de cluster
- ElasticSearch va distribuer les shards automatiquement entre les nœuds

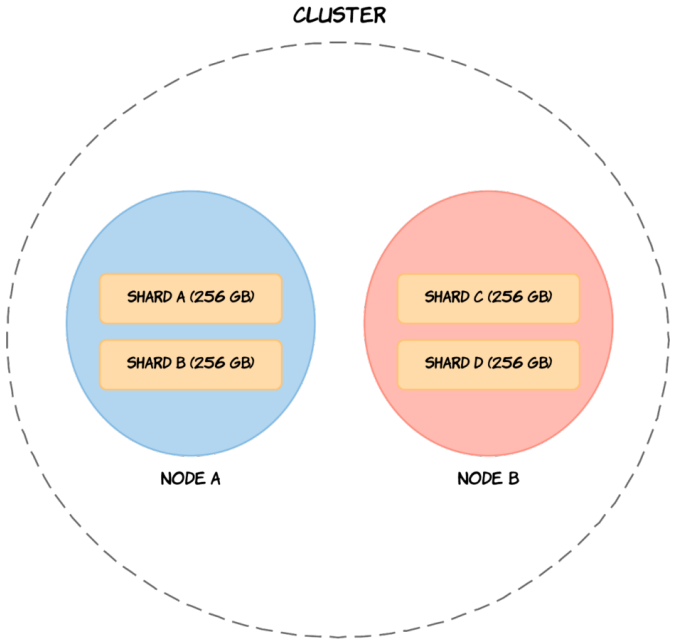


Sharding et replication

Sharding

- Si on ajoute un second nœud, on parle de cluster
- ElasticSearch va distribuer les shards automatiquement entre les nœuds

- Permet de distribuer le volume de données
- Les recherches sont distribuées entre les nodes



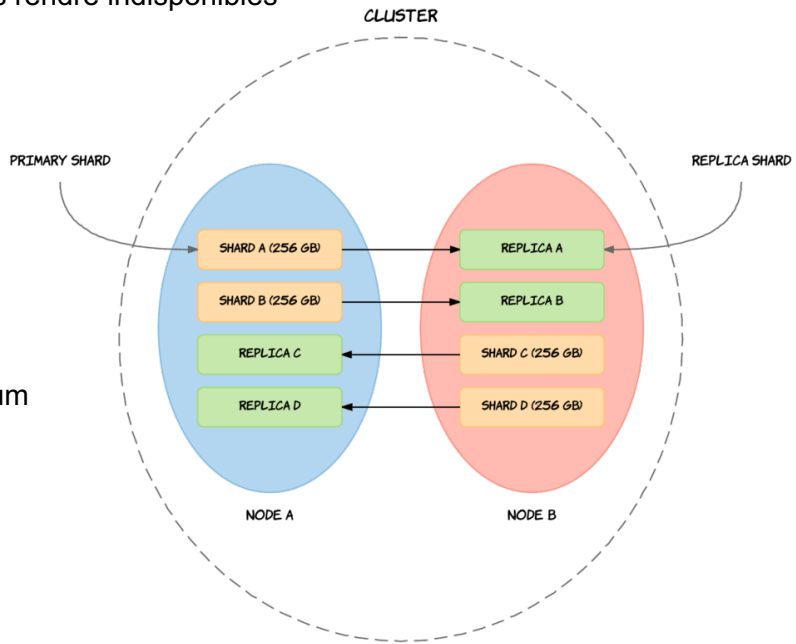
Sharding et replication

Replication

- En cas de perte d'un nœud, on ne souhaite pas perdre les données ou les rendre indisponibles
- Les shards sont donc répliqués entre les nœuds

- Avantages :
  - Rend la donnée en haute disponibilité
  - Amélioration des temps de requêtes

- Pour une haute disponibilité, il est recommandé d'utiliser 3 nœuds minimum
- Les nœuds supplémentaires sont utilisés pour la scalabilité horizontale



L’écosystème ElasticSearch - ElasticStack

ElasticStack

Cas d'utilisation

- Je dispose d'une application en architecture microservice : 15 services
- Mon services se trouve sur plusieurs serveurs : 10 serveurs
- Chaque services est installé sur plusieurs serveurs (haute disponibilité): 2 serveurs
- Chaque instance écrit des logs : 1 fichier

Combien j'ai de fichier de logs différents ?

15 * 2 * 1 = 30 fichiers différents répartis sur 10 serveurs

Je veux savoir pour la journée d'hier, combien j'ai eu d'erreurs sur mes API ?

ElasticStack

Cas d'utilisation

```
66.249.66.114 - - [24/Feb/2016:19:29:35 +0100] "GET /medias/2015/03/tuto-1-710x345.png HTTP/1.1" 304 174 "-" "Googlebot-Image/1.0"
66.249.66.112 - - [24/Feb/2016:19:36:15 +0100] "GET /medias/2015/03/tuto-15-710x267.png HTTP/1.1" 304 174 "-" "Googlebot-Image/1.0"
66.249.66.114 - - [24/Feb/2016:19:42:57 +0100] "GET /medias/2015/03/tuto-11-710x445.png HTTP/1.1" 304 174 "-" "Googlebot-Image/1.0"
66.249.66.112 - - [24/Feb/2016:19:49:35 +0100] "GET /medias/2015/03/logo-seo-camp.png HTTP/1.1" 304 173 "-" "Googlebot-Image/1.0"
66.249.66.114 - - [24/Feb/2016:19:56:15 +0100] "GET /medias/2015/03/seo-at-work-127x150.png HTTP/1.1" 304 173 "-" "Googlebot-Image/1.0"
66.249.66.112 - - [24/Feb/2016:20:02:57 +0100] "GET /medias/2015/03/tuto-3-710x290.png HTTP/1.1" 304 174 "-" "Googlebot-Image/1.0"
66.249.66.112 - - [24/Feb/2016:20:07:12 +0100] "GET / HTTP/1.1" 200 13612 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
66.249.66.110 - - [24/Feb/2016:20:09:35 +0100] "GET /medias/2015/03/tuto-14.png HTTP/1.1" 304 173 "-" "Googlebot-Image/1.0"
66.249.66.110 - - [24/Feb/2016:20:16:16 +0100] "GET /medias/2015/03/tuto-2-710x290.png HTTP/1.1" 304 174 "-" "Googlebot-Image/1.0"
66.249.66.114 - - [24/Feb/2016:20:22:58 +0100] "GET /medias/2015/03/tuto-9-710x159.png HTTP/1.1" 304 174 "-" "Googlebot-Image/1.0"
66.249.66.96 - - [24/Feb/2016:20:29:41 +0100] "GET /medias/2015/03/tuto-13-710x196.png HTTP/1.1" 304 174 "-" "Googlebot-Image/1.0"
66.249.66.94 - - [24/Feb/2016:20:36:16 +0100] "GET /medias/2015/03/tuto-10.png HTTP/1.1" 304 173 "-" "Googlebot-Image/1.0"
66.249.66.112 - - [24/Feb/2016:20:42:58 +0100] "GET /medias/2015/03/tuto-81-710x187.png HTTP/1.1" 304 174 "-" "Googlebot-Image/1.0"
66.249.66.110 - - [24/Feb/2016:20:49:36 +0100] "GET /medias/2015/03/tuto-12-710x445.png HTTP/1.1" 304 174 "-" "Googlebot-Image/1.0"
66.249.66.100 - - [24/Feb/2016:21:02:59 +0100] "GET /medias/2013/02/logo-botify.png HTTP/1.1" 304 173 "-" "Googlebot-Image/1.0"
66.249.66.114 - - [24/Feb/2016:21:09:37 +0100] "GET /medias/2013/02/majestic-seo-logo-200x68.png HTTP/1.1" 304 173 "-" "Googlebot-Image/1.0"
66.249.66.112 - - [24/Feb/2016:21:16:17 +0100] "GET /medias/2013/02/majesticseo-300x179.png HTTP/1.1" 304 173 "-" "Googlebot-Image/1.0"
66.249.66.110 - - [24/Feb/2016:21:22:59 +0100] "GET /medias/2013/02/contexte-liens.png HTTP/1.1" 304 173 "-" "Googlebot-Image/1.0"
66.249.66.114 - - [24/Feb/2016:21:29:38 +0100] "GET /medias/2013/02/puzzle-150x150.png HTTP/1.1" 304 173 "-" "Googlebot-Image/1.0"
```

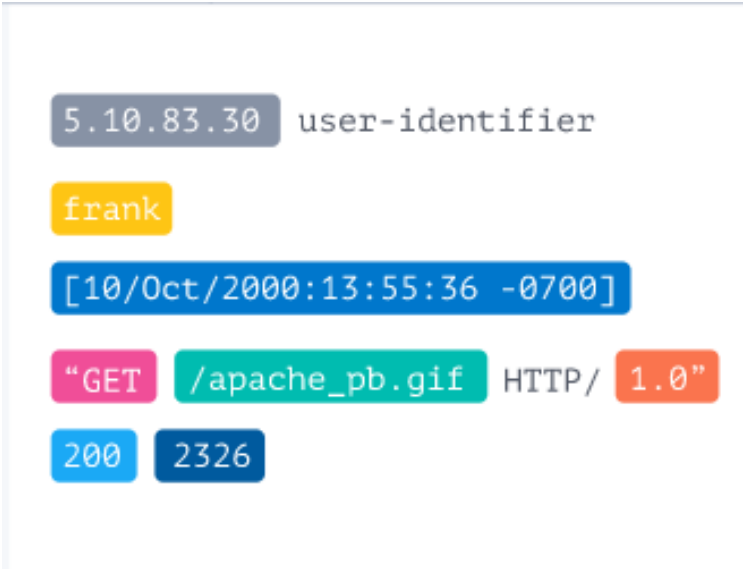
- Je veux rechercher les codes erreurs dans mes fichiers
- Je veux connaître les URLs les plus utilisées
- Je veux savoir d'où proviennent les requêtes

ElasticStack

Cas d'utilisation

- Qu'est ce qu'un fichier de logs ?

- Il s'agit d'un journal d'événements :
  - Horodatés : chaque ligne contient la date et l'heure
  - Contient des informations structurées (très rarement en JSON) ou non :
    - L'IP de l'appelant
    - Le login
    - Le code retour HTTP
    - L'URI de la requête
    - La taille de la requête ou de la réponse
    - Le temps de réponse
    - Le navigateur et système d'exploitation utilisés
    - ….



ElasticStack

Cas d'utilisation

- D'où proviennent les logs ?

- Ils peuvent provenir de différents composants de l'application :
  - Logs Système
  - Logs des bases de données
  - Logs des applications
  - Logs d'erreur
  - Logs de sécurité
  - ….

ElasticStack

Cas d’utilisation

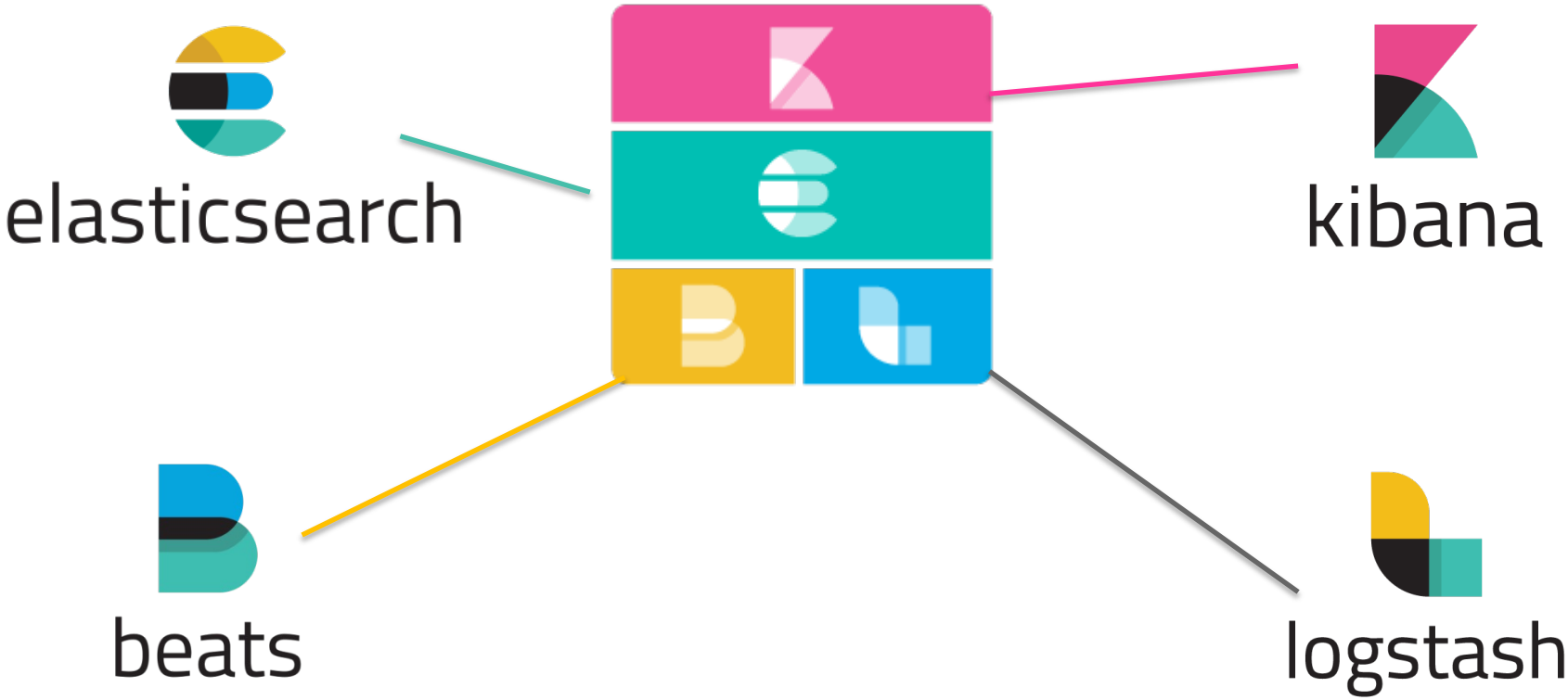
- Qu’est ce que la stack Elastic ?

- Un ensemble d’applications openSource autour d’Elasticsearch pour la manipulation de logs et de métriques à des fins d’observabilité

- Permet de centraliser l’ensemble des logs
- De les intégrer en temps réel
- De créer des graphes sur les données
- De manipuler les données
- De corréler les données

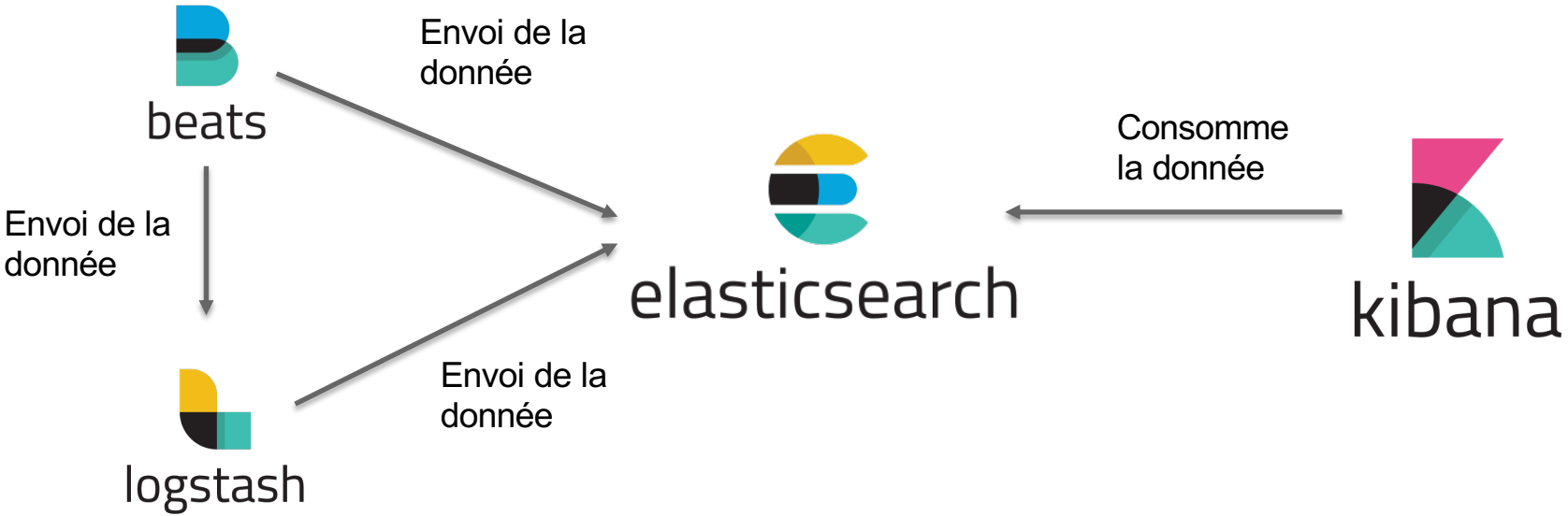
ElasticStack

Présentation



ElasticStack

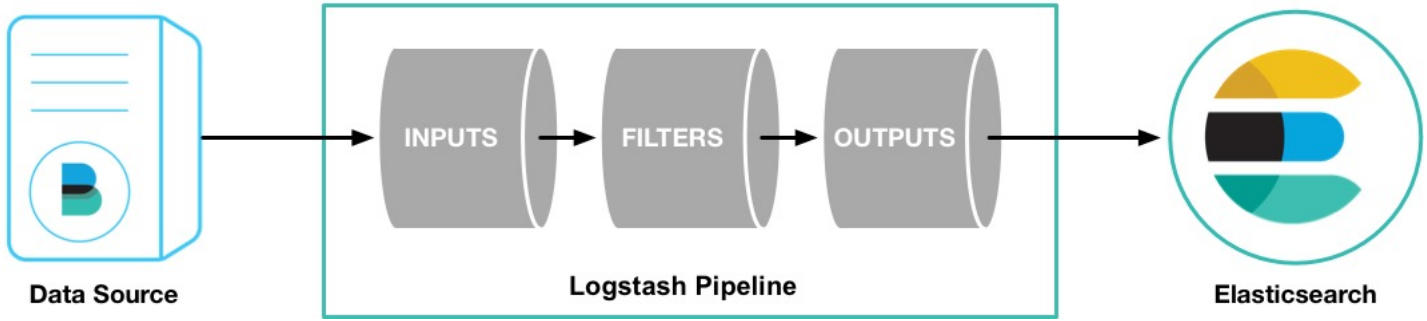
Topologie d'installation



ElasticStack

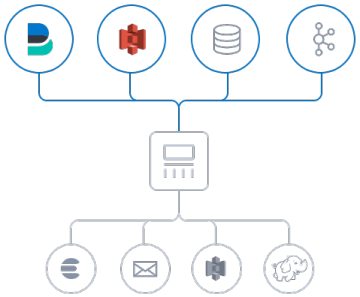
LogStash

- LogStash est un ETL :
  - **E**xtract (lecture depuis une source de données)
  - **T**ransform (filtrage / enrichissement)
  - **L**oad (envoi des données)



ElasticStack

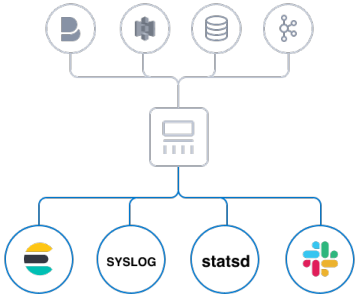
LogStash



- Extraction : Lecture de la donnée
  - De nombreux plugins de lecture de données
  - Base de données, Fichier, Rsyslog, …

- Filtrer et enrichir les données
  - Transforme des données non structrées en structurées (Champs séparés par des espaces)
  - Supprime des données non souhaitées : Données personnelles ou sans valeur
  - Calcule les coordonnées géographiques d'une IP
  - Ajoute le serveur de provenance du log
  - Formate les données : date, IP, …

- Envoyer et écrire les données :
  - De nombreux plugins disponibles
  - ElasticSearch, Syslog, Fichier, base de données, …



ElasticStack

Beats

- Un ensemble d'agents pour collecter des métriques de sources diverses.
- Se positionne en amont de LogStash ou ElasticSearch

Filebeat
Fichiers de logs
→

Metricbeat
Indicateurs
→

Packetbeat
Données réseau
→

Winlogbeat
Logs des événements Windows
→

Auditbeat
Données d'audit
→

Heartbeat
Monitoring de la disponibilité
→

Functionbeat
Agent de transfert sans serveur
→

ElasticStack

Kibana

- Interface cliente graphique de visualisation et exploration :
  - Graphiques : Camemberts, histogrammes, nuages de mots, courbes, etc.
  - Cartes géographiques
  - Indicateurs / métriques
  - Tableaux de données

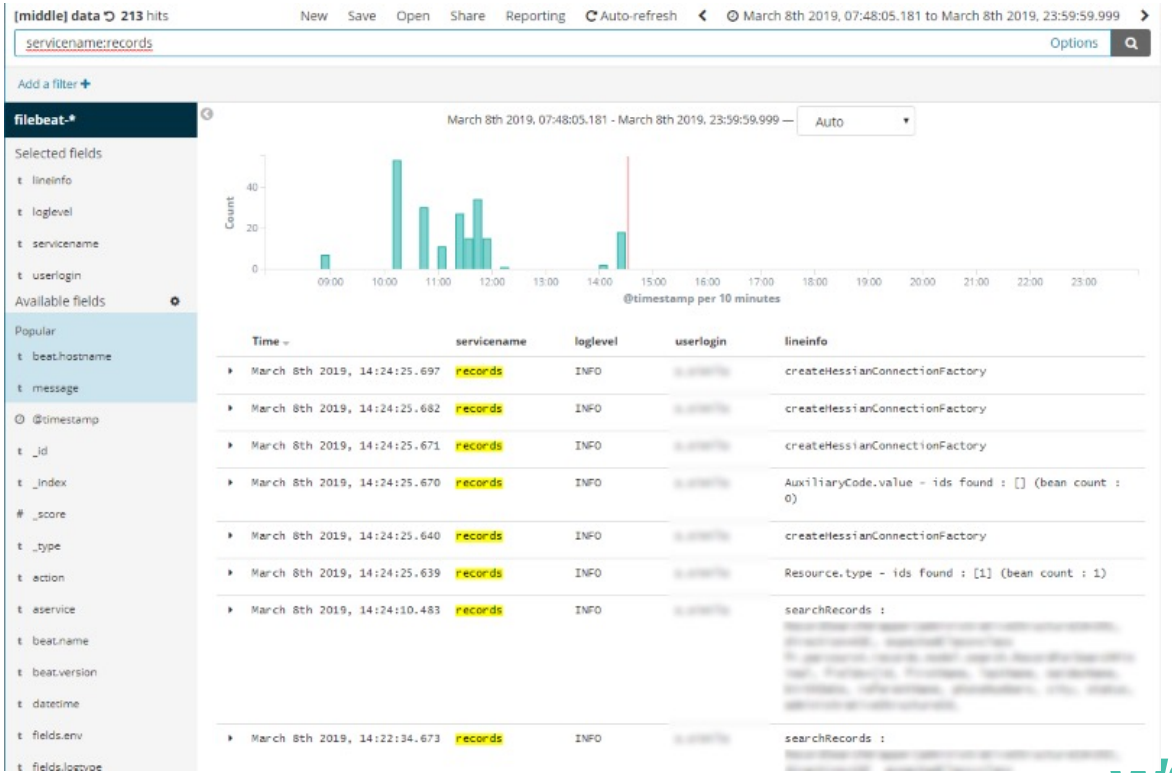
- Système de dashbord
  - Regroupe des visualisations

- Navigation des graphs :
  - Filtrer les graphs suivant une données précise : le code retour HTTP = 200
  - L'ensemble des graphs se met à jour suivant ce filtre

- Rend visuel l'analyse de logs et plus rapide :
  - Détection de problèmes : Améliore la réactivité des équipes en cas d'incident
  - Résolution de problèmes : Cible l'API qui remontent des erreurs
  - Comportement : permet de connaitre l'utilisation de l'application par les utilisateurs

ElasticStack

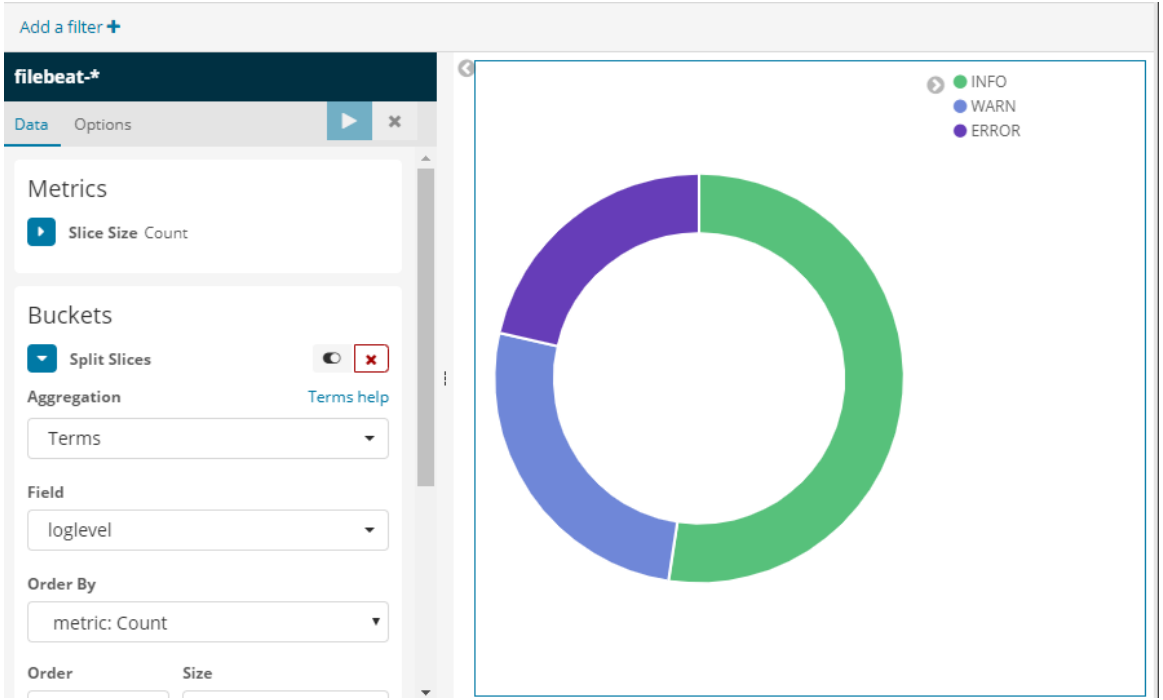
Kibana - exploration



kibana

ElasticStack

Kibana - Visualisation



ElasticStack

Kibana - dashboard

kibana

Bibliographie

Bibliographie

- https://www.gekko.fr/les-bonnes-pratiques-a-suivre-pour-developper-des-apis-rest/
- https://blog.zenika.com/2012/11/14/premiers-pas-avec-elasticsearch-partie-1/
- https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4474691-etudiez-le-fonctionnement-d-elasticsearch
- https://codingexplained.com/coding/elasticsearch/understanding-replication-in-elasticsearch
- https://codingexplained.com/coding/elasticsearch/understanding-analysis-in-elasticsearch-analyzers