Document & Index dans Elasticsearch

Index

Un **index** est l'unité principale où Elasticsearch stocke les données.
Il sert d'**espace logique** pour regrouper des documents qui ont des caractéristiques communes (ex. : ventes, utilisateurs, logs).

En formulant plus simplement : Un index, c'est comme un dossier où Elasticsearch range des documents ayant la même structure.

Chaque index a un **nom unique** (ou un **alias**) permettant de le cibler dans les requêtes de recherche, les mises à jour ou les suppressions.

Exemple :

- `clients` → contient les données sur les clients
- `ventes` → contient les transactions

Un concept proche est le **data stream** (flux de données).
C'est une forme spéciale d'index conçue pour les **données horodatées** (par exemple des logs ou métriques).
Les data streams créent et gèrent automatiquement plusieurs index « internes » pour stocker les données récentes et anciennes sans intervention manuelle.

Si vous travaillez avec des données horodatées, la solution à utiliser est **Elastic Observability**

En pratique :

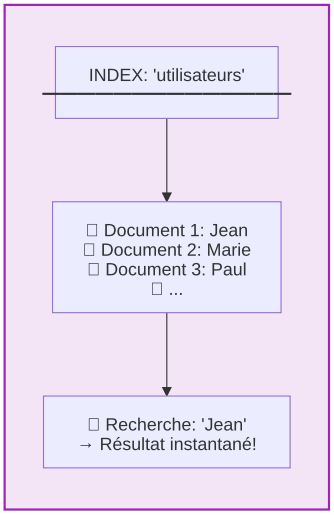
- Les **index** servent pour les données classiques (produits, utilisateurs, etc.).
- Les **data streams** sont faits pour les **données continues dans le temps** (logs, capteurs, événements).

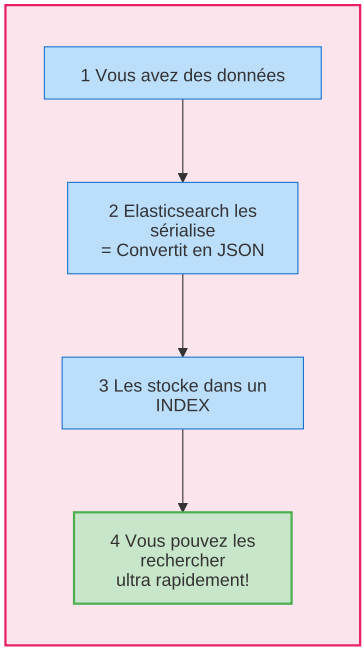
Documents et champs

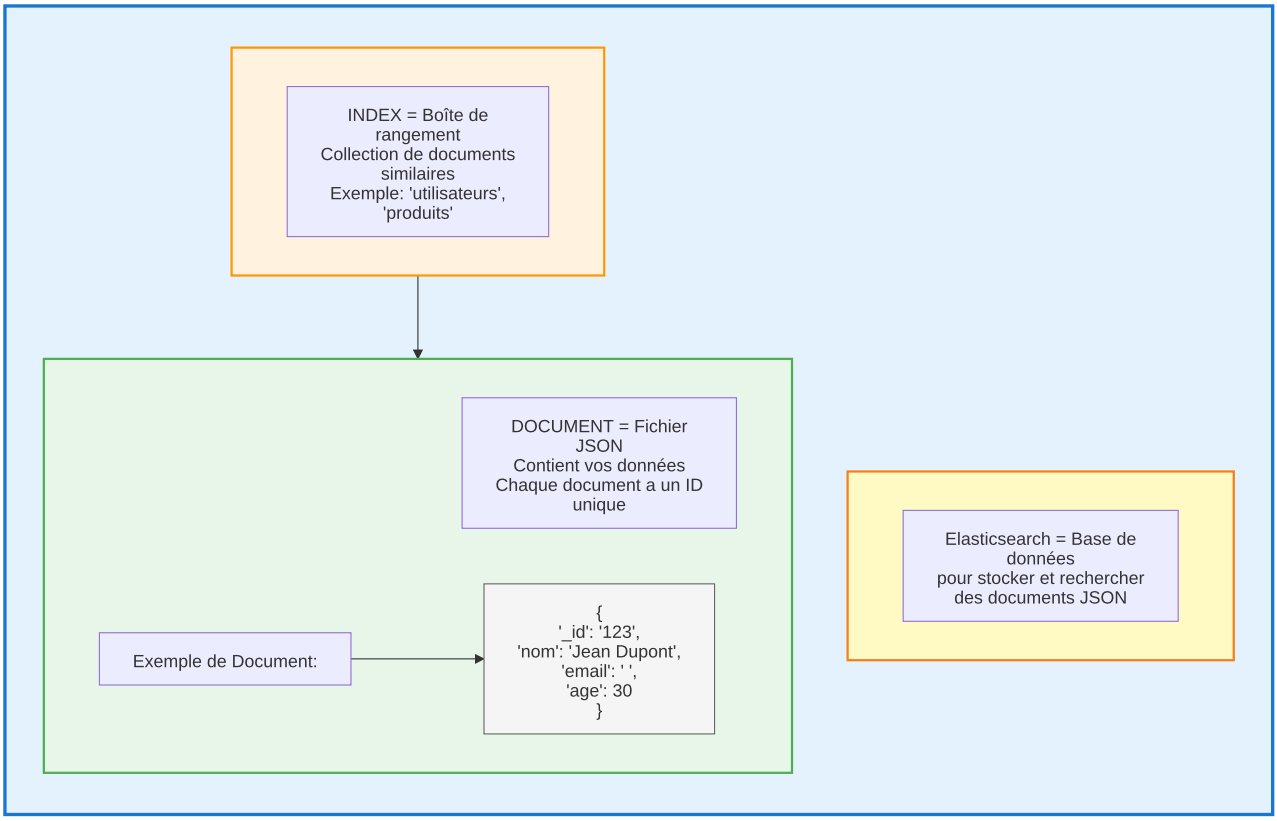
Elasticsearch sérialise et stocke les données sous forme de documents JSON. Un document est un ensemble de champs, qui sont des paires clé-valeur contenant vos données. Chaque document possède un identifiant unique, que vous pouvez créer ou laisser Elasticsearch générer automatiquement.

Exemple de document simple


```
{
  "_index": "my-first-elasticsearch-index",
  "_id": "DyFpo5EBxE8fzbb95DOa",
  "_version": 1,
  "_seq_no": 0,
  "_primary_term": 1,
  "found": true,
  "_source": {
    "email": "john@smith.com",
    "first_name": "John",
    "last_name": "Smith",
    "info": {
      "bio": "Éco-guerrier et défenseur des faibles",
      "age": 25,
      "interests": [
        "dauphins",
        "baleines"
      ]
    },
    "join_date": "2024/05/01"
  }
}
```


INDEX: 'utilisateurs'
Document 1: Jean
Document 2: Marie
Document 3: Paul
...
Recherche: 'Jean'
→ Résultat instantané!


1 Vous avez des données
2 Elasticsearch les sérialise
= Convertit en JSON
3 Les stocke dans un INDEX
4 Vous pouvez les rechercher
ultra rapidement!


INDEX = Boîte de rangement
Collection de documents similaires
Exemple: 'utilisateurs', 'produits'
DOCUMENT = Fichier JSON
Contient vos données
Chaque document a un ID unique
Exemple de Document:
{
'_id': '123',
'nom': 'Jean Dupont',
'email': ' ',
'age': 30
}
Elasticsearch = Base de données
pour stocker et rechercher
des documents JSON

Un document dans Elasticsearch contient :

1. **Les données** → les informations que vous ajoutez (ex. nom, prix, date, etc.).
2. **Les métadonnées** → des informations internes utilisées par Elasticsearch pour gérer le document.

Les **champs de métadonnées** sont des **champs système**, ajoutés automatiquement.
Ils sont toujours **préfixés par un trait de soulignement (`_`)**.

Exemples de métadonnées :

- `_index` → nom de l'index où se trouve le document
- `_id` → identifiant unique du document
- `_score` → pertinence du document dans une recherche
- `_source` → contenu original du document (les données que vous avez envoyées)

En résumé :

- Les **données** décrivent **le contenu**.
- Les **métadonnées** décrivent **le contexte et la gestion** du document.

Mappages et types de données

Chaque **index** possède un **mappage** (ou **schéma**) qui décrit la structure des documents qu'il contient.

Le mappage précise :

- le **type de données** de chaque champ (texte, nombre, date, etc.),
- la **façon dont il est indexé** (analysé pour la recherche, ou non),
- et la **manière dont il est stocké**.

---

1. Mappage dynamique

Elasticsearch **détecte automatiquement** le type des champs à partir des données reçues.

Avantage : rapide, aucune configuration.
Inconvénient : le type choisi peut ne pas convenir (ex. un code postal vu comme un nombre au lieu d'un texte).

Exemple :

```
POST clients/_doc
{
  "nom": "Durand",
  "code_postal": "75001"
}
```

→ Elasticsearch crée automatiquement un champ `code_postal` de type `integer`, alors qu'il aurait fallu `keyword`.

---

2. Mappage explicite

Vous **définissez vous-même** les types avant d'ajouter les données.

Avantage : contrôle total et cohérence des recherches.
Recommandé pour la production.

Exemple :

```
PUT clients
{
  "mappings": {
    "properties": {
      "nom": { "type": "text" },
      "code_postal": { "type": "keyword" }
    }
  }
}
```

---

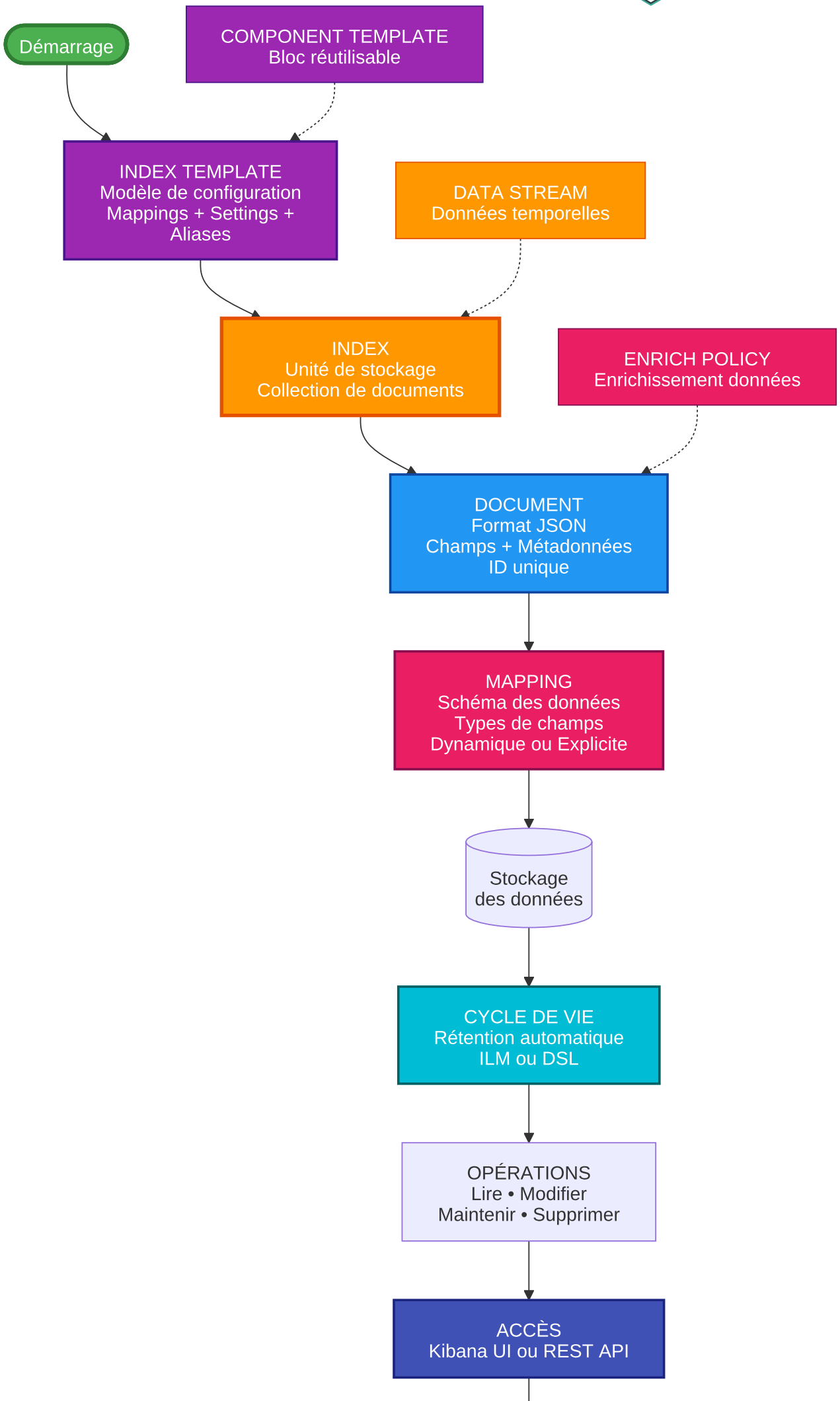
3. Combinaison des deux

Il est possible de **mixer** mappage explicite et dynamique :

- les champs connus sont définis à l'avance,
- les champs imprévus sont ajoutés automatiquement.

En résumé :

- **Dynamique** → pratique pour explorer.
- **Explicite** → fiable pour la production.


Démarrage
COMPONENT TEMPLATE
Bloc réutilisable
INDEX TEMPLATE
Modèle de configuration
Mappings + Settings +
Aliases
DATA STREAM
Données temporelles
INDEX
Unité de stockage
Collection de documents
ENRICH POLICY
Enrichissement données
DOCUMENT
Format JSON
Champs + Métadonnées
ID unique
MAPPING
Schéma des données
Types de champs
Dynamique ou Explicite
Stockage
des données
CYCLE DE VIE
Rétention automatique
ILM ou DSL
OPÉRATIONS
Lire • Modifier
Maintenir • Supprimer
ACCÈS
Kibana UI ou REST API