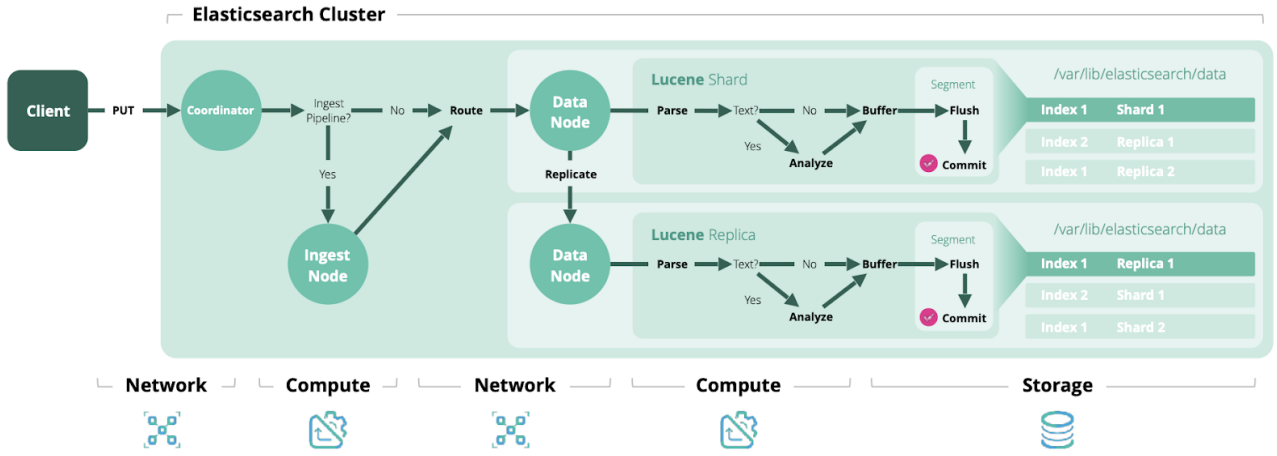
ElasticSearch

Qu’est-ce qu’un cluster ElasticSearch ?

Un cluster Elasticsearch est composé de plusieurs nœuds qui communiquent entre eux. Chaque nœud correspond à une instance d’Elasticsearch en cours d’exécution, et peut être ajouté ou retiré du cluster même lorsque ce dernier est en train de fonctionner.



À quoi sert Elasticsearch ?

La vitesse et la scalabilité d'Elasticsearch, ainsi que sa capacité à indexer de nombreux types de contenus signifient qu'il peut être employé dans différents cas d'utilisation :

- Recherche applicative
- Recherche de site web
- Entreprise Search

- Logging et analytique de log
- Indicateurs d'infrastructure et monitoring de conteneur
- Monitoring des performances applicatives
- Analyse et visualisation de données géospatiales
- L'analyse de la sécurité
- Analyse des données métier

Comment fonctionne Elasticsearch ?

Des données brutes transitent dans Elasticsearch depuis une multitude de sources, y compris des logs, des indicateurs de système et des applications web. *L'ingestion des données* est le processus qui analyse, normalise et enrichit ces données brutes avant leur *indexation* dans Elasticsearch. Une fois les données indexées dans Elasticsearch, les utilisateurs peuvent lancer des requêtes complexes à partir de leurs données et utiliser des agrégations pour récupérer des résumés complexes de leurs données. Avec Kibana, les utilisateurs peuvent créer des visualisations puissantes de leurs données, partager des tableaux de bord et gérer la Suite Elastic.

Qu'est-ce qu'un index Elasticsearch ?

Un *index* Elasticsearch est une collecte de documents en lien les uns avec les autres. Elasticsearch stocke des données sous forme de documents JSON. Chaque document met en corrélation un ensemble de *clés* (les noms des champs ou des propriétés) avec leurs valeurs correspondantes (chaînes, nombres, booléens, dates, choix de *valeurs*, géolocalisations ou autres types de données).

Elasticsearch utilise une structure de données appelée index inversé et conçue pour faire des recherches full-text très rapides. Un index inversé liste chaque mot unique qui apparaît dans n'importe quel document et identifie tous les documents dans lesquels chaque mot apparaît.

Illustration:

Un index par langue permet d'avoir la même structure avec une analyse differente :

```
PUT /blogs-en
{
"mappings": {
    "properties": {
        "title": {
```

```
            "type": "text",
                "fields": {
                    "stemmed": {
                        "type":
                            "text",
                            "analyzer": "english"
} } } } } }


PUT /blogs-fr
{
 "mappings": {
    "properties": {
        "title": {
            "type": "text",
                "fields": {
                        "stemmed": {
                            "type":
                                "text",
                                "analyzer":"french"
} } } } } }
```

À quoi sert Logstash ?

Logstash, l'un des produits de base de la Suite Elastic, sert à agréger et à traiter des données pour les envoyer dans Elasticsearch. Logstash est un pipeline côté serveur en open source destiné au Traitement des données, qui vous permet d'ingérer simultanément des données provenant de multitude de sources, puis de les transformer avant qu'elles soient indexées dans Elasticsearch.

À quoi sert Kibana ?

Kibana est un outil de visualisation et de gestion de données pour Elasticsearch qui intègre des histogrammes en temps réel, des graphes linéaires, des camemberts et des cartes. Kibana intègre également des applications comme Canvas, qui permet aux utilisateurs de créer des infographies dynamiques personnalisées basées sur leurs données, et sur Elastic Maps pour visualiser des données géospatiales.

Pourquoi utiliser Elasticsearch ?

Elasticsearch est rapide. Comme Elasticsearch est conçu d'après Lucene, il excelle en recherche full-text. Elasticsearch est également une plateforme de recherche en temps quasi réel, ce qui signifie que la latence entre

le moment d'indexation du document et la possibilité de le rechercher est très courte (généralement une seconde). Par conséquent, Elasticsearch est adapté aux cas d'utilisation urgents comme les analyses de sécurité ou le monitoring d'infrastructure.

Elasticsearch est naturellement distribué. Les documents stockés dans Elasticsearch sont distribués dans différents conteneurs appelés *partitions*, qui sont dupliqués pour intégrer des copies doublées des données en cas de défaillance matérielle. La nature distribuée d'Elasticsearch lui permet de scaler des centaines (voire même des milliers) de serveurs et gérer des pétaoctets de données.

Elasticsearch dispose d'une multitude de fonctionnalités. En plus de sa vitesse, de sa scalabilité et de sa résilience, Elasticsearch possède plusieurs fonctionnalités puissantes intégrées qui rendent le stockage et la recherche de données encore plus efficaces, comme le cumul de données et la gestion du cycle de vie des index.

Quels sont les langages de programmation pris en charge par Elasticsearch ?

Elasticsearch prend en charge une multitude de langages et les clients officiels sont disponibles pour:Java, JavaScript (Node.js), Go, .NET (C#), PHP, Perl, Python, Ruby

Quels sont les langages textes pris en charge par Elasticsearch ?

Elasticsearch prend en charge 34 langages textes allant de l'arabe au thaïlandais et fournit des analyseurs pour chacun d'eux. La liste complète est disponible dans la Documentation sur l'analyseur linguistique d'Elasticsearch. Il est possible d'ajouter un support pour des langages supplémentaires avec des plug-ins personnalisés.

Elasticsearch fournit-il des API REST ?

Oui, Elasticsearch fournit un ensemble complet et puissant d'API REST pour effectuer des tâches comme la vérification de la santé du cluster, des opérations CRUD (création, lecture, mise à jour, suppression), rechercher des opérations à partir d'indices et exécuter des opérations de recherche avancées comme le filtrage et les agrégations.

Requête DSL (Domain Specific Language)

Pour requêter vos données dans *Elasticsearch*, l'outil vous propose d'utiliser un langage spécifique basé sur du *JSON*.

Les requêtes les plus courantes sont :`**match**` : la requête renvoie les documents contenant des termes similaires au terme

recherché, mesurés par une distance de *Levenshtein*.

`term` : la requête envoie les documents contenant un terme exact dans un champs fourni. Vous pouvez utiliser requête `term` pour rechercher des documents en fonction d'une valeur précise, telle qu'un prix, un ID de produit ou un nom d'utilisateur.

`terms`: elle est semblable à une requête **`term`** et vous permet de rechercher avec plusieurs valeurs.**`prefix`**:la requête renvoie les documents commençant par la valeur fournie.

Notion de Boolean Query = une requête qui permet la **combinaison booléenne** d'autres requêtes. Chaque combinaison possède au minimum une occurrence.

Il y a 4 types d'occurrence booléenne possibles :`**must**`: opératon ET.`**must_not**`: opération NON ET.**`should`**: opération OU **`filter`**: comme le **`must`** mais le scoring sera ignoré.

Notion de Scoring : une technique permettant d'affecter un score à un document

Quelques exemples :

```
GET /club/_search
{
    "query": {
        "match": {
            "about": "Rock"
                }
},
    "sort": {
    "created": {
    "order": "desc"
        }
    }
}
```

Debug avec l'API _explain

```
GET /club/_doc/1/_explain
{
    "query": {
        "script": {
            "script": "Debug.explain(doc['age'])"
                }
            }
}
```

Déclaration du Pipeline avec un Processor :

```
PUT _ingest/pipeline/rename_foo
{
    "description": "Rename my old field",
        "processors": [
            {
                "rename": {
                    "field": "foo",
                        "target_field": "foobar"
                          }
                }
        ]
}
```

Tester le Pipeline avec document :

```
POST _ingest/pipeline/rename_foo/_simulate
{
    "docs" : [
        {
            "_source": {
                "foo": "test"
                         }
        }
            ]
}
```

Utiliser le Pipeline lors d'une indexation :

```
PUT /club/_doc/test?pipeline=rename_foo
{
    "foo": "bar"
}
```

Utiliser un GET pour récupérer un résultat :

```
GET /club/_doc/test
```

En cas d'erreur le document n'est pas indexé

```
PUT /club/_doc/test?pipeline=rename_foo
{
    "not": "bar"
}
> field [foo] doesn't exist
```

En conclusion Elasticsearch est un outil performant qui présente beaucoup de fonctionnalités et d'options, qui permet de traiter la création d'index.