

ELASTICSEARCH - ELA

```
{  
  type: "formation",  
  jours: 3,  
  objectif: "Administrer Elasticsearch"  
}
```



elastic

PRÉSENTATION

- Du formateur
 - De **DocDoku**
-

À PROPOS DE VOUS

- Expérience avec Elasticsearch
- Ce que vous attendez de la formation
- Vos projets à venir

MODALITÉS

- Horaires : 9h -> 17h30
- Pauses : 15 minutes matin et après midi
- Déjeuner : 1h30

OBJECTIFS PÉDAGOGIQUES

- Comprendre le fonctionnement d'Elasticsearch
- Evaluer les possibilités offertes par Elasticsearch
- Installer et configurer Elasticsearch
- Administrer et surveiller un cluster Elasticsearch

AGENDA

- [0] Introduction
- [1] Présentation et installation d'Elasticsearch
- [2] Fonctionnement d'Elasticsearch
- [3] Découverte et utilisation avancée de Kibana
- [4] Logstash
- [5] Configuration du cluster
- [6] Administration du cluster

INTRODUCTION



elastic

BESOINS

Nous sommes amenés à stocker de plus en plus d'informations, et avons besoin de les rechercher de manière **efficace** et **rapide**.

Les bases de données relationnelles peuvent convenir pour certaines fonctionnalités, mais peuvent être très longues à répondre pour certaines **requêtes avancées**.

Elasticsearch tente de répondre à ce besoin, tout en apportant un lot de **fonctionnalités supplémentaires**.

QU'EST-CE QU'ELASTICSEARCH ?

- Moteur de recherche et d'indexation de données
- Depuis la version 7.11, double licence : **SSPL (Server Side Public License)** ou **Elastic License 2.0 (ELv2)**
- Écrit en **Java**
- Clusterisé, sauvegarde et réplication des données
- Recherche quasi temps réel (**NRT**) (petite latence)
- Plusieurs fonctionnalités
 - Indexation
 - Recherche
 - Analyse (metrics, aggs)
 - Pertinence (scoring)

CARACTÉRISTIQUES

- Système distribué
- Architecture **REST**
- Basé sur la Apache Lucene (Apache Software Foundation)
- API disponibles pour plusieurs langages
- Format **JSON** pour l'API REST
- Pas de schéma de données
- Scalable
 - Peut tourner sur un PC portable
 - Ou sur des centaines de serveurs

ECOSYSTÈME

La société Elastic fourni en plus d'Elasticsearch :

- Kibana (visualisation)
- Logstash/Beats (ingestion)
- Agents (intégration aux applicatifs)
- Outils de monitoring
- SDK (bibliothèques pour différents langages)

HISTORIQUE

- Créé en 2004 par Shay Banon
- 0.4.0 sortie en 2010 (considérée comme la première version)
- 1.0 sortie en 2012
- 2.0 sortie en 2015
- 5.0 sortie en 2016
- 6.0 sortie en 2017
- 7.0 sortie en 2019
- 8.0 sortie en 2022
- Aujourd'hui en version **9.2.0** (23 octobre 2025)

[1] PRÉSENTATION ET INSTALLATION

- Les prérequis d'installation.
- Installation type "as a Cloud".
- La mise en œuvre d'Elasticsearch et Kibana.
- La configuration d'Elasticsearch.
- Les principes clés l'administration d'Elasticsearch.
- Le rôle de Logstash et de Kibana

PRÉREQUIS D'INSTALLATION

Java : Elasticsearch a besoin de **Java** (JRE)

Ce n'est plus nécessaire depuis la version 7. Java est maintenant embarqué avec Elasticsearch.

L'installation peut se faire de différentes façons

- Téléchargement des binaires
(<https://www.elastic.co/downloads/elasticsearch>)
- Image docker
(docker.elastic.co/elasticsearch/elasticsearch:VERSION)

HARDWARE

RAM

- Sera la première chose limitante pour de gros volumes de données
- Tris et agrégations en sont gourmands
- 64GB est la recommandation

CPU

- Utilisation peu intensive
- Préférer plus de cœurs à plus de vitesse pour paralléliser les traitements

HARDWARE

Disques

- SSD évidemment
- En général ce qui bride (entrées sorties)

Réseau

- Importance de la stabilité et de la vitesse
- Avoir peu de latence entre les différents nœuds
- Un cluster ne doit pas couvrir de trop grandes zones géographiques : « Un cluster par continent »

INSTALLATION LOCALE

Récupérer et décompresser l'archive depuis

<https://www.elastic.co/downloads/elasticsearch>

Dans le dossier décompressé on peut voir les dossiers/fichiers suivants :

```
|— bin  
|— config  
|— jdk  
|— lib  
|— LICENSE.txt  
|— logs  
|— modules  
|— NOTICE.txt  
|— plugins  
|— README.asciidoc
```


LANCEMENT DU SERVEUR

Ouvrir un terminal dans le dossier décompressé et exécuter :

```
$ bin/elasticsearch
```

Repérer la ligne de log indiquant que le serveur est démarré

```
[2025-06-17T13:20:51,272][INFO ][o.e.n.Node] [node-1] started
```

Accéder à <https://localhost:9200> avec un navigateur internet pour vérifier que le serveur répond

VÉRIFICATION DE L'INSTALLATION

Avec un client HTTP (navigateur, curl, httpie, ...), vérifier que le serveur est prêt à recevoir des données

```
$ curl https://localhost:9200
$ curl -XPUT https://localhost:9200/fruits
$ curl -XPUT https://localhost:9200/legumes
$ curl https://localhost:9200/_cat/indices
```

avec l'option `-ku elastic:<password>`

Nous utiliserons cette syntaxe pour la suite

```
GET /
PUT fruits
PUT legumes
GET _cat/indices
```

CONFIGURATION STATIQUE

Dans le dossier **config**, on trouve plusieurs fichiers de configuration

```
config
├── elasticsearch.keystore
├── elasticsearch.yml
├── jvm.options
├── log4j2.properties
├── role_mapping.yml
├── roles.yml
├── users
└── users_roles
```

- Paramètres Elasticsearch : **elasticsearch.yml**
- Options JVM : **jvm.options**
- Paramètres Log4j2 : **log4j2.properties**

PARAMÈTRES ELASTICSEARCH

- Les paramètres par défaut sont déjà bien définis
- Peu de configuration nécessaire pour une utilisation basique
- Utilisation du format YAML

Syntaxe YAML, basé sur les **indentations**, notation avec "."
conseillée.

Exemple:

```
node:
  name: mon-serveur
node.name: mon-serveur
# ceci est un commentaire
```

PARAMÈTRES BASIQUES

Nom du cluster : doit être **le même** pour toutes les instances elastic

Nom du noeud : pour différencier les instances

```
cluster.name: mon-cluster  
node.name: mon-serveur-1  
path.data: /chemin/vers/donnees  
path.logs: /chemin/vers/logs  
network.host: 192.168.0.1  
http.port: 9200
```

FONCTIONS AVANCÉES DE LA CONFIGURATION

Il est possible d'utiliser des **variables d'environnement** directement dans le fichier YML

```
node.name: ${HOSTNAME}
```

Il est aussi possible de renseigner une valeur au lancement

```
node.name: ${prompt.text}
```

OPTIONS JVM

Le fichier **jvm.option** contient les **options** de lancement de la **JVM** pour Elasticsearch. Elles sont visibles dans les logs de démarrage :

```
[2025-06-17T13:20:33,749][INFO ][o.e.n.Node] [node-1] JVM arguments [-Xshare:auto, -Des.networkaddress.cache.ttl=60, -Des.networkaddress.cache.negative.ttl=10, -XX:+AlwaysPreTouch, -Xss1m, -Djava.awt.headless=true, -Dfile.encoding=UTF-8, -Djna.nosys=true, -XX:-OmitStackTraceInFastThrow, -XX:+ShowCodeDetailsInExceptionMessages, -Dio.netty.noUnsafe=true, -Dio.netty.noKeySetOptimization=true, -Dio.netty.recycler.maxCapacityPerThread=0, -Dio.netty.allocator.numDirectArenas=0, -Dlog4j.shutdownHookEnabled=false, -Dlog4j2.disable.jmx=true, -Djava.locale.providers=SPI,COMPAT, -Xms1g, -Xmx1g, -XX:+UseG1GC, -XX:G1ReservePercent=25, -XX:InitiatingHeapOccupancyPercent=30, -Djava.io.tmpdir=/tmp/elasticsearch-7201219294924291084, -XX:+HeapDumpOnOutOfMemoryError, -XX:HeapDumpPath=data, -XX:ErrorFile=logs/hs_err_pid%p.log, -Xlog:gc*,gc+age=trace,safepoint:file=logs/gc.log:utctime,pid,tags:filecount=32,filesize=64m, -XX:MaxDirectMemorySize=536870912, -Des.path.home=/home/morgan/Downloads/elasticsearch-7.7.1, -Des.path.conf=/home/morgan/Downloads/elasticsearch-7.7.1/config, -Des.distribution.flavor=default, -Des.distribution.type=tar, -Des.bundled_jdk=true]
```

MÉMOIRE JVM

Par défaut, Elasticsearch prend la moitié de la RAM physique disponible.

Pour modifier la RAM allouée, éditer **jvm.options** :

```
# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms32g
-Xmx32g
```

32G est la taille maximale recommandée.

Toujours donner au **maximum** la moitié de la RAM physique disponible.

PARAMÈTRES LOG4J2

Configuration avancée de Log4j 2 :

<https://www.elastic.co/guide/en/elasticsearch/reference/current/logging.html>

- Rotation des logs
- Compression et archivage
- Niveaux de logs
- Emplacement des logs
- Formatage des messages

PRINCIPES D'ADMINISTRATION

- Outils de **monitoring**
- S'assurer de la **santé** d'un ou des clusters
- S'assurer de la santé des nœuds
- Modification de la **répartition de charge**
- Monitoring des **performances des requêtes**
- Ajout de mappings sur les documents
- Configuration fine des index

SANTÉ DU CLUSTER

Une API de vue d'ensemble de la santé du cluster.

```
GET _cluster/health
```

```
{
  "cluster_name" : "elasticsearch",
  "status" : "yellow",
  "timed_out" : false,
  "number_of_nodes" : 1,
  "number_of_data_nodes" : 1,
  "active_primary_shards" : 1,
  "active_shards" : 1,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 1
}
```

CODES COULEUR

Green : tout va bien, tous les réplicas sont **alloués** correctement

Yellow : des réplicas sont manquants (**non alloués**)

Red : il y a au moins un **shard primaire** qui est manquant, et les réplicas ne sont pas disponibles. Perte de données probable, ou corruption. Pas d'indexation possible.

ETAT DU CLUSTER

```
GET _cluster/state
```

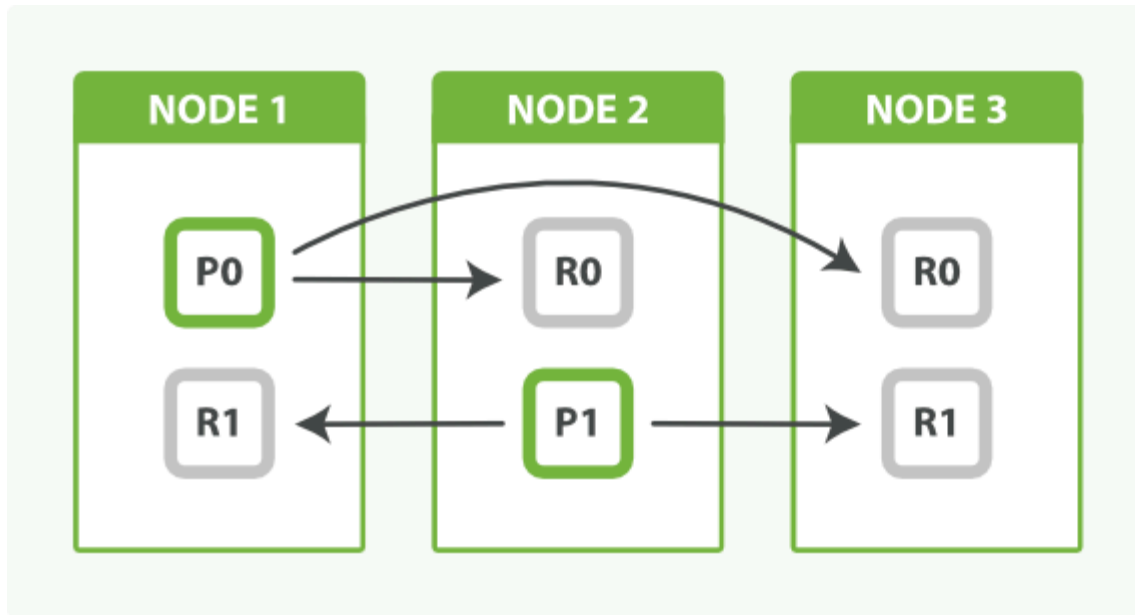
Renvoie absolument toutes les informations du cluster

- Etat des nœuds
- Configuration des nœuds
- Etat des shards
- Paramètres et mapping des indexes
- Paramètres de sécurité
- Paramètres de routing
- Corbeille des indexes

DISTRIBUTION DES DONNÉES

Les **documents** (données) contenus dans les **index** (tables), sont distribués dans des "**shards**". Un shard peut être **primaire**, ou **réplica** (copie d'un primaire).

Exemple : 3 noeuds, 2 shards, 2 replicas par shard

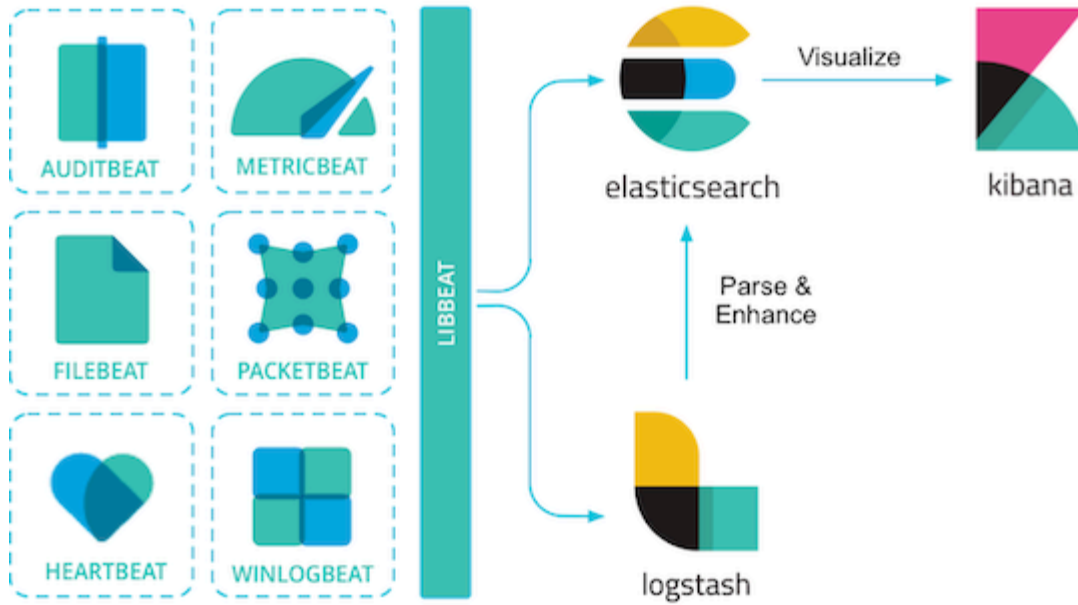


PRODUITS ANNEXES

Beats : collecte des données

Logstash : ingestion de données

Kibana : visualisation des données



LOGSTASH

Agent permettant de lire de multiple sources de données, puis de les envoyer dans Elasticsearch.

Temps réel : dès qu'une nouvelle ligne est insérée dans un fichier, Logstash la lit et l'envoie en quelques millisecondes.

Installé en général à l'extérieur du cluster.

Communique avec Elasticsearch via l'**API REST**

KIBANA

Serveur de visualisation des données issues d'Elasticsearch

Permet de visualiser sous forme de **graphiques**, **tableaux**, **cartes**, n'importe quelle donnée contenue dans les index.

Les graphiques peuvent se rafraichir en **temps réel**.

Gestion d'**espaces**, permettant de limiter les visualisations à différents utilisateurs.

Communique avec Elasticsearch via l'**API REST**

INSTALLATION DE KIBANA

La société Elastic propose plusieurs façons d'installer Kibana

- Binaires
- Paquets linux
- Image Docker

Installation locale :

- Télécharger la version souhaitée en fonction du système (<https://www.elastic.co/downloads/kibana>)
- Décompresser l'archive, ouvrir un terminal dans le dossier décompressé, et exécuter la commande :

```
$ bin/kibana
```

[2] FONCTIONNEMENT D'ELASTICSEARCH

- Présentation d'Apache Lucene.
- Architecture as a Cloud et concepts clés (cluster, node, etc.).
- Le format JSON. L'API REST.
- Mapping et type de données.
- Recherches simples et composées.
- Ingestion des données et stockage de données.

ARCHITECTURE

- Un moteur d'**indexation**
- Un moteur de **recherche**
- Une gestion **automatique** de la **répartition** de charge et de la **réplication** des données
- Une API REST pour contrôler le tout
- Pas d'interface graphique !

MOTEUR D'INDEXATION ET DE RECHERCHE

Apache Lucene

- Projet **open source**
- Ecrit par Doug Cutting en 2001
- Ecrit en **Java**, disponible sous forme de JAR
- Complètement **agnostique** de votre application
- Pas de dépendances
- Moteur d'indexation et de recherche
- Manque d'une API REST (ce qu'apporte Elasticsearch)

APACHE LUCENE

- Accepte **tout type de données**, tout type de format
- Peut donc indexer différents contenus
 - Pages web
 - Contenu et métadonnées de fichiers
 - Champs textes
 - Nombres
 - Etc...

FONCTIONNALITÉS DE LUCENE

- Les différents types de query (exact, partial, range, prefix, etc..)
- Les filtres
- Les analyzers
- Indexation incrémentale
- Score et pertinence
- Tris (attention à la RAM !!!)
- Filtres de sécurité
- Système de cache

CORRESPONDANCE SQL- ELASTICSEARCH

SQL	NoSQL
Serveur de base de données	Node
Table	Index
Row	Document
Column	Field

CONCEPTS CLÉS

Cluster : Est un ensemble de serveurs (nodes) connectés entre eux

Node : Est une instance de serveur Elasticsearch, contient des shards et répliques

Shard : Partitionne les collections de documents

Replica : Réplique les shards dans différents noeuds

RÉPARTITION DES NŒUDS D'UN CLUSTER

Chaque nœud communique avec le **nœud maître**

Avoir le moins de latence possible entre les différents nœuds du cluster

« Un cluster par continent »

« Quelle taille doit faire mon cluster ? » « Ça dépend ! »

DIMENSIONNER UN CLUSTER

Dépend de beaucoup de paramètres

- De la donnée
- De la charge de votre application
- Des performances attendues
- De la taille des documents
- Du nombre de documents

COMMENT BIEN PARTITIONNER SON CLUSTER ?

L'unité atomique de scalabilité est le « **shard** »

Les shards peuvent être déplacés, mais pas divisés

Plus le shard est petit, plus il est facile de le copier / déplacer

Mais... pas de limite technique pour leur taille

Le shard devient trop gros ?

- Ajouter plus de hardware (plus de RAM/CPU/disques)
- Ajouter plus de machines
- Reconstruire l'index avec plus de shards

API REST

Elasticsearch impose son propre langage de communication
(**QueryDSL**)

Domain Specific Language

Basé sur le format **JSON** (standard)

Les échanges se font principalement via **HTTP** et l'API REST exposée

PRINCIPALES API

La « single » API

- Index API
- Get API
- Update API
- Delete API

La « bulk » API

- Multi Get API
- Bulk API
- Delete By Query API
- Update By Query API
- Reindex API

CREATION D'UN INDEX

Creation d'un **index** nommé « employés », paramètres par défaut

```
PUT employes
```

Création d'un **index** « fruits », avec des paramètres différents

```
PUT fruits
{
  "settings" : {
    "number_of_shards" : 2,
    "number_of_replicas": 1
  }
}
```

Ici, « settings » est un mot-clé du QueryDSL

INSERTION DE DONNÉES

Création d'un document représentant un employé

Sans spécifier l'ID

```
POST /employe/_doc
{
  "name" : "Alice"
}
```

En spécifiant l'ID

```
PUT /employe/_doc/12345
{
  "name" : "Alice"
}
```

Si l'index n'existe pas il sera créé (paramètre par défaut).

RETOUR DE L'INDEXATION

Elasticsearch renvoie des données relatives à l'insertion

```
{
  "_shards" : {          # Informations de réplication
    "total" :           2, # Nombre de copies lancées
    "failed" :           0, # Nombre de copies ko
    "successful" : 2     # Nombre de copies ok
  },
  "_index" : "employe", # Index dans lequel la donnée est insérée
  "_type" : "_doc",    # Type du document inséré
  "_id" : "12345",     # Id du document inséré
  "_version" : 1,      # Version du document inséré
  "created" : true,     # Créé (true) ou modifié (false)
  "result" : "created"
}
```

INSERTION OU MODIFICATION

Utiliser la méthode **POST** permet de générer des identifiants, mais ne se préoccupe pas des doublons.

Préférer la méthode **PUT** pour l'unicité des données.

Le champ **version** est incrémenté à chaque modification, même si le document n'a pas changé. Pour s'en prémunir :

```
PUT employe/_doc/12345?op_type=create
{
  "name" : "Alice"
}
```

Code d'erreur 400 si le document existe déjà.

LECTURE DE DOCUMENTS

API GET

```
GET employe/_doc/12345
```

Elasticsearch renvoie le **document** et les **métadonnées**

```
{
  "_index" : "employe",
  "_type" : "_doc",
  "_id" : "12345",
  "_version" : 1,
  "found": true,
  "_source" : { # Contenu du document
    "name" : "Alice",
    "birthday" : "1987-04-20T12:00:00",
    "favorite_color": "cyan"
  }
}
```

LECTURE DE DOCUMENTS (SOURCE)

API GET

```
GET employe/_source/12345
```

Elasticsearch renvoie le document sans les métadonnées

```
{  
  "name" : "Alice",  
  "birthday" : "1987-04-20T12:00:00",  
  "favorite_color": "cyan"  
}
```

TEST D'EXISTENCE

Méthode **HEAD**

Allège la réponse, seulement les **headers** sont échangés.

```
HEAD employe/_doc/12345  
200 OK
```

Code HTTP **404** si la resource n'existe pas

```
HEAD employe/_doc/99999  
404 NOT FOUND
```

SUPPRESSION D'UN DOCUMENT

API DELETE

```
DELETE employe/_doc/12345
```

Elasticsearch renvoie des données relatives à la suppression

```
{
  "_index": "employe",
  "_type": "_doc",
  "_id": "12345",
  "_version": 1,
  "result": "deleted",
  "_shards": { # Shards affectés
    "total": 2,
    "successful": 1,
    "failed": 0
  }
}
```

MODIFICATION PARTIELLE STATIQUE

API update

```
POST employe/_update/12345
{
  "doc" : { "name" : "Lucie" },
  "detect_noop": true
}
```

L'option **detect_noop** permet de ne pas monter la version du document si aucun changement n'est apporté.

MODIFICATION PARTIELLE SCRIPTÉE

Permet de modifier une valeur à l'aide d'un **script**, idéal pour incrémenter un champ par exemple.

```
POST employe/_update/12345
{
  "script" : {
    "source": "ctx._source.age += 1",
    "lang": "painless"
  }
}
```

Référence du langage painless :

<https://www.elastic.co/guide/en/elasticsearch/painless/master/index.html>

OPÉRATIONS EN MASSE

Bulk API

Permet de faire **plusieurs actions** en une seule fois.

```
action_and_meta_data
optional_source
action_and_meta_data
optional_source
....
action_and_meta_data
optional_source
```

Chaque ligne est un JSON, la première ligne contient la directive avec des options, la seconde est un document (donnée)

INSERTIONS AVEC BULK API

Contenu d'un fichier « employees.bulk »

```
{ "index": { "_index" : "employee", "_id" : "1" } }  
{ "name": "rob", "favorite_color": "red" }  
{ "index": { "_index" : "employee", "_id" : "2" } }  
{ "name": "anne", "favorite_color": "pink" }  
{ "index": { "_index" : "employee", "_id" : "3" } }  
{ "name": "steeve", "favorite_color": "green" }
```

Envoi du fichier à Elasticsearch avec curl

```
$ curl -s -H "Content-Type: application/json"  
  -XPOST localhost:9200/_bulk --data-binary "@employees.bulk";
```

MODIFICATION PAR REQUÊTE

Permet de modifier un ou plusieurs documents à l'aide d'une **query**.

```
POST employe/_update_by_query
{
  "script" : {
    "source": "ctx._source['age'] = 26",
    "lang": "painless"
  },
  "query": {
    "term": {
      "name": "alice"
    }
  }
}
```

SUPPRESSION PAR REQUÊTE

Permet de supprimer un ou plusieurs documents à l'aide d'une **query**.

```
POST employe/_delete_by_query
{
  "query": {
    "term": {
      "name": "alice"
    }
  }
}
```

POINTS D'ENTRÉE DE RECHERCHE

Il est possible de rechercher dans **un ou plusieurs** index

Mot clé : `_all`, possibilité d'utiliser un wildcard « `*` »

```
GET fruits,legumes/_search?q=name:tomate
GET _all/_search?q=name:Alice
GET _search?q=name:Alice
GET logs-*/_search?q=country:france
```

RECHERCHE DE DOCUMENTS

2 solutions principales

- Recherche par « query params »
- Recherche avec « request body »

```
GET /employee/_search?q=name:Alice
```

```
GET /employee/_search
{
  "query": {
    "match" : {
      "name" : {
        "query" : "Alice"
      }
    }
  }
}
```

RECHERCHE QUERY PARAMS

Exemple

```
GET logs/_search?q=(country:russia OR country:brazil) AND date>2020
```

- Intuitif (mots clés AND,OR,NOT)
- Concis
- Peut être mis en cache (URL)

En revanche

- Ne permet pas de faire des requêtes complexes et avancées.

RECHERCHES QUERYDSL

Format JSON

- Plus lisible que les recherches « inline »
- « Leaf query clauses » : clauses **simples** permettant de rechercher dans un champ donné (match, term, range, wildcard, prefix, fuzzy...)
- « Compound query » : clauses **composées** d'autres clauses composées, ou de clauses simples (bool, dis_max...) : permettent de combiner les requêtes

<https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html>

TERM QUERY

Recherche d'un terme **exact** (pas d'analyse)

```
GET blog/_search
{
  "query": {
    "term": {
      "title": {
        "value": "elasticsearch"
      }
    }
  }
}
```

MATCH QUERY

Recherche d'un terme avec analyse

```
GET blog/_search
{
  "query": {
    "match": {
      "title": {
        "query": "Elasticsearch"
      }
    }
  }
}
```

RANGE QUERY

Permet d'encadrer des valeurs

```
GET employe/_search
{
  "query": {
    "range": {
      "age": {
        "gte" : 18,
        "lte" : 22
      }
    }
  }
}
```

Mots-clés : gt(>), lt(<), gte(>=), lte(<=)

WILDCARD QUERY

Le caractère * signifie "n'importe quelle chaine de caractère"

Le caractère ? signifie "n'importe quel caractère"

```
GET employe/_search
{
  "query": {
    "wildcard": {
      "user": {
        "value": "Ali*"
      }
    }
  }
}
```

AUTRES TYPES DE REQUÊTES

Elasticsearch offre un grand nombre de type de requêtes, certaines sont spécialisées sur des termes (term-level), d'autres spécialisées sur du contenu textuel (full-text)

On peut aussi trouver des requêtes spécialisées :

- Geo queries (recherches sur des points, zones, box)
- Script query (recherche scriptée)
- More-like-this query (recherche de similarités)
- Percolate (recherche inversée)

RE-INDEX API

API permettant de copier les données d'un index dans un nouvel index.

Pratique pour re-dimensionner un index avec plus (ou moins) de shards.

```
POST _reindex
{
  "source" : {
    "index" : "index1"
  },
  "dest" : {
    "index" : "index2"
  }
}
```

LES CONTEXTES DE RECHERCHE

Elasticsearch attribue un score aux documents recherchés, et les classe dans l'ordre du meilleur score par défaut.

Il y a 2 contextes :

- **Query context** (calcul du score)
- **Filter context** (élimination)

Query context : comment le document correspond à ma requête ?

Filter context : le document correspond oui ou non à ma requête ?

REQUÊTES ET CONTEXTES

Exemple de requête composée

```
GET employe/_search
{
  "query": {
    "bool": {
      "must": [
        { "wildcard": { "name": { "value": "Ali*" }}}
      ],
      "filter": [
        { "range": { "age": { "gte": 30 }}}
      ]
    }
  }
}
```

must (must_not, should) : calcul du score, **filter** : élimination

REQUÊTES BOOL QUERY

```
GET employe/_search
{
  "query": {
    "bool": {
      "must": [..., ...],
      "must_not": [..., ...],
      "should": [..., ...],
      "filter": [..., ...],
    }
  }
}
```

- **must** : obligatoire
- **must_not** : ne doit pas
- **should** : peut avoir (pratique pour booster)
- **filter** : oui ou non

LES DATATYPES

Parmi les plus utilisés

Chaines : text, keyword

Numerique : long, integer, short, byte, double, float, half_float, scaled_float

Dates : date, date_nanos

Booléens : boolean

Mais aussi... : integer_range, float_range, long_range, double_range, date_range, ip_range, nested, geo_point, geo_shape, ip, completion, alias...

DYNAMIC MAPPING

Elastic devine pour nous les types de données lors de la découverte d'un champ.

```
PUT employe/_doc/12345
{
  "name" : "Alice",
  "age" : 20,
  "last_connection": "2020-06-12T20:15:00"
}
```

- name : text + keyword (2 fois stocké)
- age : long
- last_connection : date (format ISO)

LECTURE DES MAPPINGS EXISTANTS

```
GET employe/_mapping
```

```
"age" : {  
  "type" : "long"  
},  
"last_connection" : {  
  "type" : "date"  
},  
"name" : {  
  "type" : "text",  
  "fields" : {  
    "keyword" : {  
      "type" : "keyword",  
      "ignore_above" : 256  
    }  
  }  
}  
}
```

ANALYSE ET STOCKAGE

Les champs de type « text » sont **analysés** avant d'être stockés (analyser par défaut).

L'analyser par défaut enlève la **punctuation**, les **majuscules**.

Si on stocke le code "FACT.123456-CLIENT-1", voici ce qui est stocké :

```
[ "fact", "123456", "client", "1" ]
```

Les analyseurs nettoient, découpent et stockent un **tableau de mots** (tokens)

ANALYSE LINGUISTIQUE

Si on stocke la chaîne : "Where is Brian ? In the kitchen." avec un analyseur anglais, voici ce qui est réellement stocké :

```
[ "where", "brian", "kitchen" ]
```

Si on stocke la chaîne "J'aime les légumes, mais je préfère les boulettes." avec un analyseur français, voici ce qui est réellement stocké :

```
[ "aime", "legum", "prefer", "boulet" ]
```

En fonction du type de recherche, les mots recherchés sont aussi analysés et confrontés aux mots stockés.

ANATOMIE D'UN ANALYSER

Un analyser est composé de 3 composants

- « **Character filter** » : Supprime ou modifie certains caractères (entités HTML, caractères Unicode, etc...)
- « **Tokenizer** » : Découpe un texte en plusieurs chaînes de caractères selon un délimiteur défini.
- « **Token filters** » : Modifie / supprime certaines chaînes de caractère obtenus depuis le Tokenizer

Elasticsearch fourni un nombre conséquent d'analyseurs. Il est aussi possible de composer son analyser personnalisé.

<https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis.html>

INGESTION ET PIPELINES

Il est possible d'**enrichir** les données avant l'insertion.

Exemples :

- Ajouter un champ
- Transformer une date
- Obtenir la géolocalisation en fonction d'une IP
- Encoder/Décoder
- Désérialiser du JSON
- ...

CRÉATION D'UNE PIPELINE

Utilisation de l'API pipeline

```
PUT _ingest/pipeline/urgent
{
  "description" : "Ajoute un tag urgent",
  "processors" : [
    {
      "set" : {
        "field": "tag",
        "value": "urgent"
      }
    }
  ]
}
```

On note que **processors** est un tableau.

LES PROCESSORS

Une trentaine de processors sont disponibles dans le coeur d'Elasticsearch. Parmi les plus utilisés :

- Append
- CSV
- Date
- Grok
- GeoIP
- UserAgent

Les processors sont exécutés dans l'ordre de définition.

<https://www.elastic.co/guide/en/elasticsearch/reference/current/ingest-processors.html>

PIPELINE À LA SUITE D'UNE PIPELINE

Possibilité d'enchaîner les pipelines avec le processor "pipeline"

```
{
  "description" : "ma pipeline",
  "processors" : [
    {...},
    {
      "pipeline" : {
        "name": "autre-pipeline"
      }
    }
  ]
}
```

CAS D'ERREURS

Une pipeline peut rencontrer des erreurs (IP non valide, JSON mal formé, etc...)

Dans ce cas, il est possible d'effectuer une ou plusieurs actions :

- Ignorer
- Refuser l'indexation
- Enrichir autrement (ajout d'un tag erreur par exemple)

L'erreur peut être gérée sur chaque processor, ou bien au niveau global de la pipeline.

GESTION DES ERREURS (1-3)

Au niveau d'un processor :

```
{
  "rename" : {
    "field" : "user",
    "target_field" : "user_name",
    "on_failure" : [
      {
        "set" : {
          "field" : "error",
          "value" : "error while renaming"
        }
      }
    ]
  }
}
```

GESTION DES ERREURS (2-3)

Au niveau global de la pipeline

```
{
  "description" : "Ma pipeline ...",
  "processors" : [ ... ],
  "on_failure" : [
    {
      "set" : {
        "field" : "_index",
        "value" : "error-{{ _index }}"
      }
    }
  ]
}
```

Ici, les documents en erreur seront stockés dans un autre index (error-nom_index)

GESTION DES ERREURS (3-3)

Ignorer les erreurs avec **ignore_failure** :

```
{
  "description" : "Ma pipeline",
  "processors" : [
    {
      "rename" : {
        "field" : "user",
        "target_field" : "user_name",
        "ignore_failure" : true
      }
    }
  ]
}
```

Le document sera stocké sans transformation

UTILISATION DES PIPELINES

2 façons d'utiliser une pipeline :

- De manière automatique
- En spécifiant la pipeline lors d'une insertion

DEFAULT PIPELINE

Configurable au niveau de l'index

```
PUT nouvel-index
{
  "settings" : {
    "default_pipeline" : "ma-pipeline"
  }
}
```

Chaque document inséré dans cet index sera enrichi via la pipeline "ma-pipeline"

CHOIX D'UNE PIPELINE À L'INSERTION

Pour chaque insertion :

```
PUT employe/_doc/12345?pipeline=ma-pipeline
{
  "name" : "Alice",
  "age" : 30
}
```

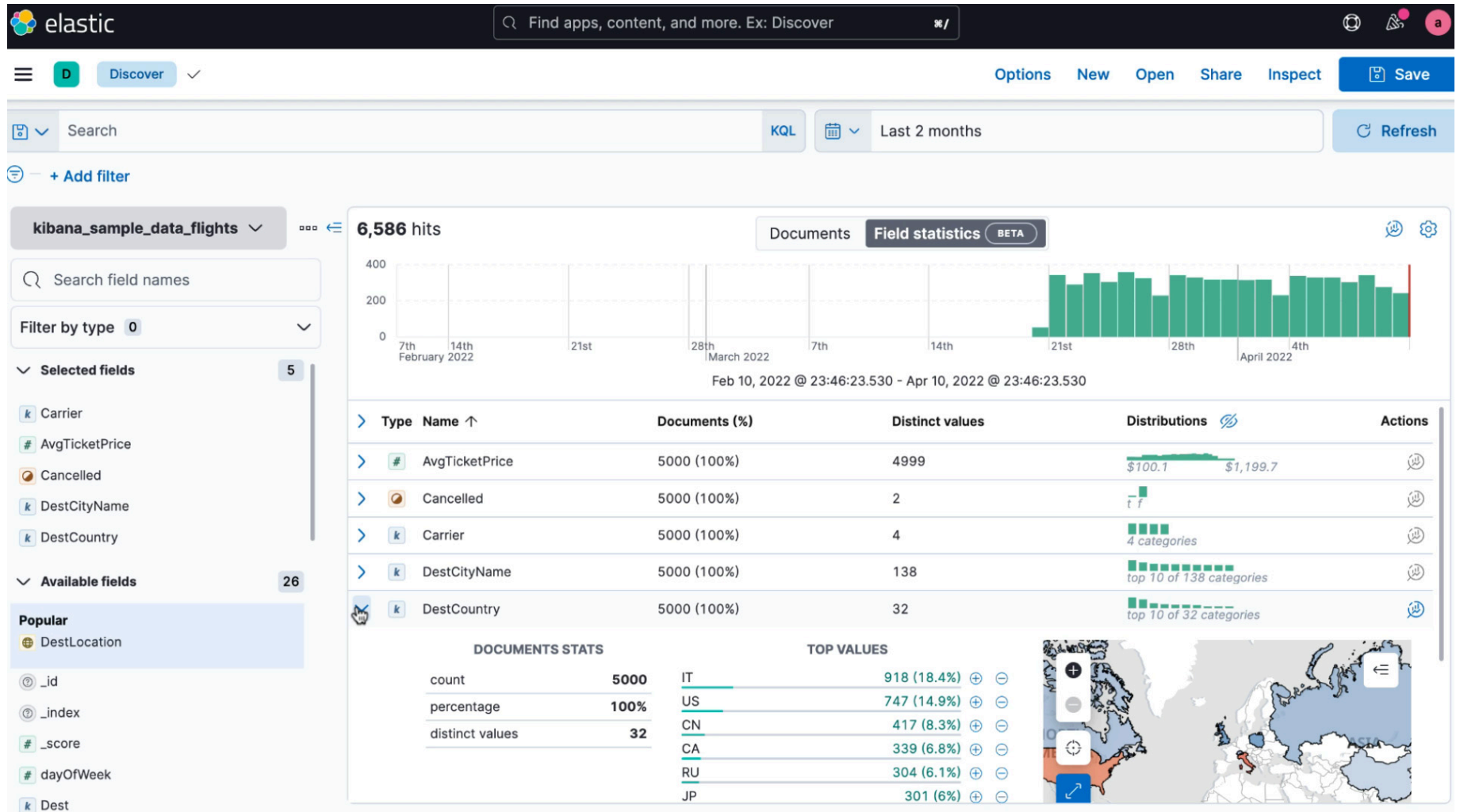
[3] VISUALISER LES DONNÉES AVEC KIBANA

- Présentation de l'interface
- Le contexte de travail : data view
- Le filtre temporel
- L'exploration de données avec Discover
- Visualize Library et la création graphiques
- Tableaux de bord et gestion des objets
- Alerting

POINTS CLÉS :

- Produit de la « ElasticStack »
- Application web pour explorer, découvrir, naviguer dans les données Elasticsearch
- Tableaux de bord personnalisés
- Graphiques (camemberts, histogrammes, etc.)
- Console Elasticsearch QueryDSL
- Import / export de tous les objets
- Index propre pour tout stocker « .kibana »
- Accès à Kibana (par défaut <http://localhost:5601>)

APERÇU DE L'INTERFACE



4 ZONES PRINCIPALES

« **Sidebar** » : permet d'accéder aux différents modules et plugins. Les principaux : Discover, Visualize Library, Dashboard et Management

« **Searchbar** » : permet de rechercher avec la syntax KQL (AND, OR avec completion)

« **Toolbar** » : permet d'agir sur tous les objets (New, Save, Open, Share, Inspect, Edit...)

« **Timefilter** » : permet de déterminer le filtre temporel à appliquer aux données (Last 15 minutes, Last 5 years, To... From....)

INDISPENSABLE POUR DÉMARRER DANS KIBANA

Création d'une « data view » : **contexte de travail** filtrant les données sur un ou plusieurs index (« index1 » ou « index* »).

Choix du champ temporel : active le « time filter » (date obligatoire)

La « data view » est ensuite utilisable au sein des différents modules pour filtrer / contextualiser les données

METTRE EN PLACE DES VUES

Depuis le menu « **Discover** » :

- Exploration des données par « data view »
- Création de « discover session », c.a.d des vues en tableau, triées (« sort ») et filtrées « filter »
- Possibilité de sauvegarder les « discover session »

METTRE EN PLACE DES VISUALISATIONS

Depuis le menu « **Visualize Library** » :

- Création de métriques ou agrégations (Sum, Count, Min, Max...)
- Création de graphiques (Area, Line, Vertical Bar, Pie....)
- Possibilité de sauvegarder les « visualization »

CRÉATION D'UN GRAPHIQUE (1-3)

1ère étape : **les métriques**

- Choix d'un type de calcul : moyenne, somme, min, max, nombre, etc..
- Plusieurs métriques peuvent être affichées sur le même graphique selon son type.

CRÉATION D'UN GRAPHIQUE (2-3)

2ème étape : **les buckets** ou **breakdowns**

- Permettent de découper le graphique selon différentes méthodes. Découpage en X, découpage des séries, découpages en plusieurs graphiques
- Possibilité de grouper / découper par un terme donné, une plage numérique, une plage de dates, un interval de temps, des coordonnées géographiques, etc.

CRÉATION D'UN GRAPHIQUE (3-3)

3ème étape : **les axes et options**

- Choix des échelles, couleurs, légendes
- Couleurs selon un seuil
- Options spécifiques au type de graphique (barres stackées, camembert plein, couleurs de zones, ...)

METTRE EN PLACE DES TABLEAUX DE BORD

Depuis le menu « **Dashboard** » :

- Création de « dashboard » en ajoutant des « discover session » ou des « visualization »
- Possibilités de partager avec ces tableaux de bord
- Tous ces objets sont sauvegardés dans l'index « .kibana » et accessibles dans le module « Management / Kibana / Saved Objects »
- Import / export également possibles de tous ces objets au format JSON

GESTION DES OBJETS SAUVEGARDÉS (1-2)

- Interface de gestion
- Inclus tous les objets (tableaux, graphiques, dashboards, ...)
- Possibilité d'exporter/importer
- Gestion des dépendances entre objets

GESTION DES OBJETS SAUVEGARDÉS (2-2)

Via l'API de Kibana

Lister les objets d'un type donné :

```
$ curl localhost:5601/api/saved_objects/_find?type=dashboard
```

```
[
  {
    "type": "dashboard",
    "id": "e4ad8a40-c04b-11ea-86ef...",
    "attributes": {
      ...
    }
  }
]
```

EXPORT VIA L'API

Passage en paramètre de la liste d'objets à exporter

```
$ curl -X POST api/saved_objects/_export \  
-H 'kbn-xsrf: true' \  
-H 'Content-Type: application/json' -d '  
{  
  "objects": [  
    {  
      "type": "dashboard",  
      "id": "e4ad8a40-c04b-11ea-86ef..."  
    }  
  ]  
}'
```


ALERTING

Principe :

- **Condition** : qu'est-ce qui doit être détecté ?
- **Programmation** : quand/à quel intervalle doit on effectuer le test ?
- **Action** : que faire lors de la détection ?

Disponible avec la **license basique** depuis peu.

Usage limité : possibilité d'exécuter l'action **index** seulement (ajout de données dans un index)

ALERTING, MISE EN OEUVRE

Prérequis : définir une clé de chiffrement (32 caractères minimum)

Dans **kibana.yml**

```
xpack.encryptedSavedObjects.encryptionKey: "....."
```

Puis, redémarrer Kibana et accéder à Stack Management > Alerts and actions

DÉFINITION DE L'ALERTE (1-3)

Choisir un type, un nom, une fréquence de détection

Create alert

Name

Tags (optional)

Check every ?

minute



Notify ?

Only on status change



DÉFINITION DE L'ALERTE (2-3)

Choisir un index, une condition

Index threshold



1

Select an index

INDEX logs-apache

WHEN count()

OVER all documents

2

Define the condition

IS ABOVE 100000 FOR THE LAST 1 minute

DÉFINITION DE L'ALERTE (3-3)

Choisir une action (index ou log serveur en version basique)

Actions

▼  alerts-index 

Index connector

Add new

alerts-index ▼

Document to Index 

1 ▼ {

2 "id" : "{{alertId}}",

3 "name" : "{{alertName}}"

4 }

Plusieurs champs sont disponibles (dates, valeurs, nom/id de l'alerte, ...)

[4] LOGSTASH

- Présentation
- Les sources
- Les filtres
- Les sorties
- Configuration et lancement

LOGSTASH

« Centralisez, transformez et stockez vos données »

- Programme d'**ingestion** de données
- Plusieurs sources supportées
- Parfait pour débiter avec de la donnée
- Envoie les données collectées dans Elasticsearch (mais pas que)

Peut être utilisé sans Elasticsearch : on peut imaginer lire des tweets, vérifier s'il contient le mot "lance-le-café", et lancer une commande vers sa cafetière connectée...

LES SOURCES

Une multitude de **plugins**

- Twitter
- Github
- Databases : redis, sqlite, ... Elasticsearch !
- Les logs machine
 - Intéressants pour visualiser en quasi temps réel des informations sur la machine

<https://www.elastic.co/guide/en/logstash/current/input-plugins.html>

LES FILTRES

Pour chaque événement de la source (nouveau flux réseau, nouvelle ligne dans un fichier, etc...) on peut appliquer un ou plusieurs **filtres**.

- Encodage/décodage
- GeoIP
- Découper un CSV
- Scripts Ruby
- ...

<https://www.elastic.co/guide/en/logstash/current/filter-plugins.html>

LES SORTIES

Une fois filtré, l'événement peut être envoyé.

- Notifications
- Email
- Elasticsearch évidemment
- Lancement de scripts
- Bases de données, mongodb, redis, mysql, ...
- Syslog
- ...

<https://www.elastic.co/guide/en/logstash/current/output-plugins.html>

FICHERS DE CONFIGURATION

Pour définir une **chaîne de traitement**, Logstash se base sur des fichiers de configuration. La syntaxe lui est propre.

Un fichier de configuration décrit la source, les filtres et la sortie.

```
input {  
  # Plugin d'entrée  
}  
  
filter {  
  # Filtres  
}  
  
output {  
  # Sorties  
}
```

MULTIPLS CONFIGURATIONS

Logstash peut traiter plusieurs chaînes à la fois. Dans ce cas on utilisera les **pipelines** de logstash.

Fichier **pipelines.yml**

```
- pipeline.id: tweets-vers-cafe
  path.config: "/chemin/vers/config1.logstash"
  pipeline.workers: 2
- pipeline.id: csv-vers-elastic
  path.config: "/chemin/vers/config2.logstash"
  pipeline.workers: 4
```

Possibilité de configurer finement les ressources allouées à chaque chaîne de traitement

<https://www.elastic.co/guide/en/logstash/current/logstash-settings-file.html>

LANCEMENT DE LOGSTASH

Utilisation d'un seul fichier de config :

```
bin/logstash -f /chemin/vers/fichier-config.logstash
```

Dans le cas de l'utilisation du fichier de pipelines, lancer simplement :

```
bin/logstash
```

Ces commandes ne rendent pas la main, logstash va **scruter** les sources définies.

FILE D'ATTENTE PERSISTANTE



- Configuration globale dans **logstash.yml**
- **Par défaut** : *queue.type: memory*
 - événements stockés uniquement **en mémoire**
 -  risque de perte si Logstash ou machine tombe
- **Option persistante** : *queue.type: persisted*
 - événements écrits **sur disque** avant traitement
 - assure la **durabilité** (redémarrage ou panne)

```
queue.max_bytes : taille max (défaut = 1 GB)
queue.checkpoint.writes : fréquence de sauvegarde (défaut = 1024)
```

[5] CONFIGURATION DU CLUSTER

- Configuration du cluster Elasticsearch
- Traitement des gros volumes
- Configuration des nodes
- Configuration des indexes
- Les traitements avant indexation avec les pipelines
- Gestion des templates
- Gestion des alias

CLUSTER NAME

-  Nommer son cluster
 - pour éviter bien des problèmes
 - souvent la première chose à faire
-  Ne pas utiliser le « discovery » automatique
 - scan des nœuds sur la même plage réseau
 - fonctionnalité pratique mais **dangereuse**
 - plus disponible dans les nouvelles versions

 **déclaration manuelle** des différents nœuds (IP ou nom)

NODE NAME

Configuration statique (elasticsearch.yml)

- Permet de repérer les machines
- Inclus dans les réponses de l'API
- Par défaut => **Hostname** de la machine

Pour le changer :

```
node.name: node1
```

PARAMÈTRES DU SYSTÈME

Quelques propriétés de base à bien vérifier

- Emplacement des données et des logs
- HEAP size (32GB maximum)
- Nombre de fichiers ouverts
- Mémoire virtuelle
- Désactiver le swap !

SWAP

- opération one-shot

```
$ swapoff  
$ sysctl -w vm.max_map_count=262144
```

- pour le faire à chaque démarrage

```
# Créer un fichier de conf Elasticsearch  
sudo nano /etc/sysctl.d/99-elasticsearch.conf  
  
# Ajouter :  
vm.max_map_count=262144  
  
# Appliquer les changements ou redémarrer  
sudo sysctl -p /etc/sysctl.d/99-elasticsearch.conf
```

ROLE DU NOEUD

6 roles sont disponibles

- essentiels :
 - **Master**
 - **Data**
 - **Ingest**
- optionnels :
 - Machine learning
 - Transform
 - Remote cluster client

ROLE MASTER

Les noeuds master sont les "**cerveaux**" du cluster

Un seul noeud est **master actif**

Ils sont "**éligibles**" à devenir master actif en cas de panne du noeud master actif

Ce sont ces noeuds qui peuvent être interrogés pour les recherches

Un noeud est master par défaut. Pour le désactiver :

```
node.master: false
```

ROLE DATA

Les noeuds data stockent les **shards**

Ces serveurs doivent être dimensionnés correctement (RAM, disques)

Ce sont ceux qui "**calculent**" et "**stockent**"

Un noeud est data par défaut. Pour le désactiver :

```
node.data: false
```

ROLE INGEST

Les noeuds ingest permettent de **transformer** les données en entrée

Pratique pour **enrichir** avant l'indexation

Un noeud est ingest par défaut. Pour le désactiver :

```
node.ingest: false
```

ROLE MACHINE LEARNING

Disponible seulement en version Platinum

Prend en charge les requêtes et les jobs de machine-learning

Pour l'activer sur un noeud :

```
xpack.ml.enabled: true  
node.ml: true
```


ROLE TRANSFORM

Permet de générer des **pivots**.

Les pivots sont principalement des **métriques** issues d'un index (sommations, moyennes, etc...) qui sont insérées dans un nouvel index.

Par défaut, un noeud data est un noeud transform.

Un noeud seulement master ne peut pas être transform.

Pour le désactiver

```
node.transform: false
```

ROLE REMOTE CLUSTER CLIENT

Ce role permet à un noeud de se **connecter** à un **autre cluster** en mode client.

Activé par défaut

Etabli une connection **unidirectionnelle** vers un cluster distant

Permet de la **réplication**, ou de la **recherche**.

Pour désactiver cette fonctionnalité

```
node.remote_cluster_client: false
```

DIMENSIONNEMENT DU CLUSTER

Les questions à se poser !

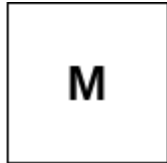
- Comment assurer la **charge** des requêtes HTTP ?
- Comment assurer la **sauvegarde** des données ?
- Comment assurer de la haute **disponibilité** ?
- Comment avoir le minimum de latence entre les nodes ?
- Comment coordonner le tout ?

RÉPARTITION DES NOEUDS

Il faut au minimum **3 noeuds master**

Ces noeuds peuvent être data et ingest

Il est possible ensuite d'ajouter des noeuds data pour assurer la puissance de calcul/stockage



... ..



DATACENTER

Il est possible de **répartir** les noeuds dans 2 datacenter
(avoir au minimum 3 master dans chaque)

Cela permet une meilleure résilience aux **pannes réseau /
électriques**

Très haute disponibilité

DÉCOUVERTE DES NOEUDS

Un **mécanisme** de **découverte** des noeuds est implémenté dans Elasticsearch.

Depuis la version 2.0, cette découverte est automatisée sur l'interface locale de la machine.

Pratique pour tester la formation d'un cluster **sur la même machine** (scan auto des plages 9200-9300).

SEED HOSTS

Déclaration manuelle des noeuds master

```
discovery.seed_hosts:  
  - 192.168.1.10:9300  
  - 192.168.1.11  
  - seeds.mydomain.com  
  - [0:0:0:0:0:ffff:c0a8:10c]:9301  
cluster.initial_master_nodes:  
  - master-node-a  
  - master-node-b  
  - master-node-c
```

Tous les noeuds doivent avoir cette même configuration

SHARDING DES INDEX

Pour répartir les données d'un index sur les différents noeuds, on peut jouer sur le **nombre de shards**.

Ce nombre de shards **ne peut pas être changé** une fois l'index créé. Il faudra ré-indexer les données dans un nouvel index si on veut augmenter/réduire ce nombre.

```
PUT nouvel-index
{
  "settings" : {
    "number_of_shards" : 5
  }
}
```


NOMBRE DE SHARDS

Combien de shards par index ?

- **+** shards **+** écritures peuvent être parallélisées
- **+** shards **-** de documents y seront stockés
 - attention au scoring
 - mais plus facile sera leur réplication / parcours

Préconisation Elasticsearch :

- Maximum **20 shard par GB de heap size** (Exemple : 600 shards maximum pour un node avec 30GB)

TEMPLATES D'INDEX

Possibilité de définir des **settings** et **mappings** par **défaut** pour les futurs indices, de manière automatique.

Possibilité de découper les templates en "component templates" (briques ré-utilisables).

CRÉATION D'UN TEMPLATE POUR LES FUTURS INDICES

Permet de définir les settings et mappings de haut niveau

Le pattern "*" correspond à tout nouvel index

```
PUT _index_template/template_general
{
  "index_patterns": ["*"],
  "template": {
    "settings": {...},
    "mappings": {...}
  }
}
```

CRÉATION D'UN TEMPLATE

Ce template sera appliqué si on crée un nouvel index appelé "logs-apache-2025-06"

```
PUT _index_template/template_logs
{
  "index_patterns": ["logs-apache-*", "logs-ngnix-*"],
  "template": {
    "settings": {...},
    "mappings": {...}
  }
}
```

LES COMPONENT TEMPLATES

Briques **réutilisables** dans des templates

```
PUT _component_template/component_1
{
  "template": {
    "settings": {...},
    "mappings": {...}
  }
}
```

```
PUT _index_template/template_1
{
  "index_patterns": ["logs-*"],
  "composed_of": ["component_1", "component_2", ...]
}
```

LES ALIAS

Les alias d'index facilitent la gestion et l'écriture des requêtes

Ne nécessite pas de changer la configuration des applicatifs

```
POST /_aliases
{
  "actions" : [
    {
      "add" : {
        "index" : "index-1",
        "alias" : "production"
      }
    }
  ]
}
```

ALIAS SCRIPTÉS

Possibilité d'embarquer un **filtre** dans l'alias

But : offrir des **vues** sur les données

```
POST /_aliases
{
  "actions" : [
    {
      "add" : {
        "index" : "logs-russie",
        "alias" : "logs",
        "filter" : { "term" : { "country" : "russia" } }
      }
    }
  ]
}
```

[6] ADMINISTRATION DU CLUSTER

- Surveillance du cluster
- La répartition de charge
- L'allocation des nodes
- Alias d'index
- Propriétés "Transient" et "Persistent"
- Principes de la sauvegarde et la restauration
- Sécurité
- Cycle de vie des indexes et rétention

SURVEILLANCE DU CLUSTER

Les points à **surveiller** :

- La santé
- Le nombre de noeuds
- La charge globale
- La disponibilité des noeuds

Ces données sont **collectées par Kibana** et stockées dans les indices `.monitoring-*` si le self monitoring est activé

SURVEILLANCE DES MÉTRIQUES SYSTÈME

A surveiller principalement :

- Utilisation de la **RAM**
- Utilisation du **CPU**
- **Charge** du système
- **Espace disque**
- **Réseau**

Les données système sont envoyés dans Elasticsearch via "Metricbeat" ou "Elastic Agent"

<https://www.elastic.co/guide/en/beats/metricbeat/current/index.html>

SURVEILLANCE DES MÉTRIQUES DES NOEUDS

Les points à surveiller :

- RAM
- **Répartition de la charge**
- Répartition des données
- Disponibilité
- Nombre de requêtes

Ces données sont automatiquement collectées par Kibana quand le self monitoring est activé

SURVEILLANCE DES MÉTRIQUES DES INDICES

Données des index à surveiller :

- Latence d'indexation
- Latence de recherche
- Taux d'indexation/recherche
- Nombre de documents
- **Taille des shards**

Ces données sont automatiquement collectées par Kibana quand le monitoring est activé

OUTILS DE SURVEILLANCE

- **MetricBeats** (à déployer sur les machines) : remontent les métriques système dans Elasticsearch
- **Kibana** : écrans de monitoring intégrés, collecte automatiquement les données de monitoring
- **Elasticvue** : bonne vision de la répartition des données (se connecte à Elasticsearch via son API REST)
- Autres produits de monitoring facilement couplables à Elasticsearch : Nagios, Shinken, Grafana

RÉPARTITION DE LA CHARGE

Elasticsearch gère de lui même la répartition en distribuant de **manière paire** les shards sur les noeuds disponibles.

Cependant, après une reprise d'un noeud, ce dernier peut avoir seulement des réplicas.

Il est possible de répartir les données manuellement.

ANALYSE DE LA RÉPARTITION

API de détails des shards

Permet d'afficher la **répartition des shards** des différents indices

```
GET _cat/shards?v
```

index	shard	prirep	state	docs	store	ip	node
index1	1	p	STARTED	0	130b	127.0.0.1	node-1
index1	1	r	UNASSIGNED				
index1	0	p	STARTED	0	130b	127.0.0.1	node-1
index1	0	r	UNASSIGNED				
index2	2	p	STARTED	0	159b	127.0.0.1	node-1
index2	1	p	STARTED	0	159b	127.0.0.1	node-1
index2	0	p	STARTED	1	6.9kb	127.0.0.1	node-1

ACTIVATION/DESACTIVATION BALANCING AUTOMATIQUE

Paramètre du cluster

```
PUT _cluster/settings
{
  "persistent": {
    "cluster.routing.allocation.enable": "all"
  }
}
```

- **all** : (default) tous les shards
- **primaries** : seulement primaires
- **new_primaries** : shards primaires des nouveaux indices
- **none** : aucun

EXCLUSION D'UN NODE

Avant une opération de maintenance

```
PUT  /_cluster/settings
{
  "persistent": {
    "cluster.routing.allocation.exclude._name": "node-name-to-drain"
  }
}
```

RÉPARTITION SELON CRITÈRE (1-2)

Dans le cas de plusieurs datacenters, il faut **répliquer** les shards d'un datacenter dans un **autre datacenter**.

La première étape consiste à définir des **paramètres personnalisés** dans la configuration des noeuds.

```
# Noeuds du datacenter n°1  
node.attr.dc=DC1  
# Noeuds du datacenter n°2  
node.attr.dc=DC2
```

node.attr permet de définir les paramètres personnalisés.

RÉPARTITION SELON CRITÈRE (2-2)

Préciser à Elasticsearch sur quel critère se baser :

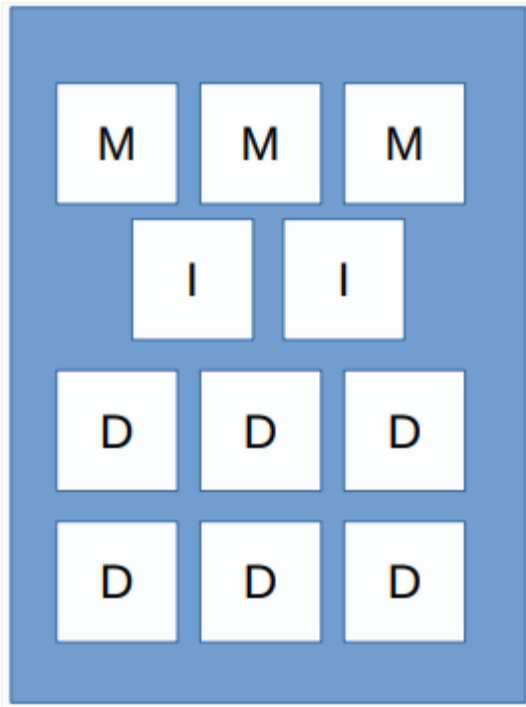
```
cluster.routing.allocation.awareness.attributes: dc
```

Choisir les valeurs de critère qui ne doivent pas partager les réplicas

```
cluster.routing.allocation.awareness.force.dc.values: DC1,DC2
```

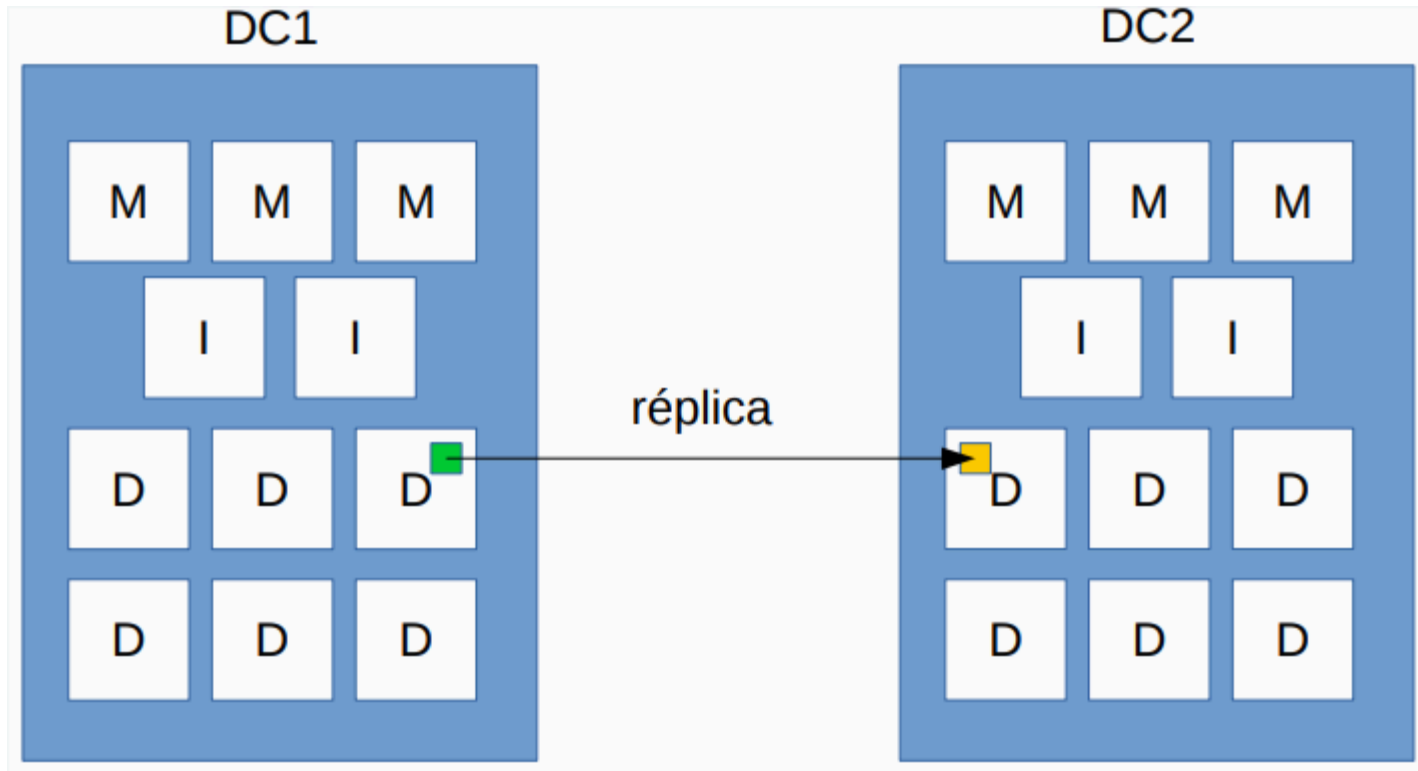
ARCHITECTURE HAUTE DISPONIBILITÉ

Possibilité d'ajouter des noeuds Data à la volée pour plus de puissance



ARCHITECTURE TRÈS HAUTE DISPONIBILITÉ

Réplication dans l'autre datacenter



PROPRIÉTÉS TRANSIENT ET PERSISTENT

La configuration du cluster peut se faire à plusieurs niveaux

- Transient (effectif jusqu'au redémarrage du cluster)
- Persistent (résiste au redémarrage du cluster)
- Statique (fichier elasticsearch.yml)
- Settings par défaut

La **précédence** est l'ordre énoncé ci-dessus

PROPRIÉTÉ TRANSIENT

Via l'API settings

```
PUT /_cluster/settings
{
  "transient" : {
    "indices.recovery.max_bytes_per_sec" : "50mb"
  }
}
```

PROPRIÉTÉ PERSISTENT

Via l'API settings

```
PUT /_cluster/settings
{
  "persistent" : {
    "indices.recovery.max_bytes_per_sec" : "50mb"
  }
}
```


SUPPRIMER UN SETTING TRANSIENT OU PERSISTANT

Via l'API settings

```
PUT /_cluster/settings
{
  "persistent" : {
    "indices.recovery.max_bytes_per_sec" : null
  }
}
```

Toute une plage :

```
PUT /_cluster/settings
{
  "persistent" : {
    "indices.recovery.*" : null
  }
}
```

SAUVEGARDE DES DONNÉES

Pourquoi faire des sauvegardes ?

La **réplication** ne joue t-elle pas ce rôle ?

=> **Non** car elle ne nous prémunit pas des **erreurs sur les données**

Attention aux incompatibilités des versions !

Possibilités de stockage : Filesystem classique, HDFS (Hadoop), Microsoft Azure, Amazon S3, Google Cloud Storage

SAUVEGARDE : ÉTAPE 1/3

La première étape consiste à déclarer le « repository » qui hébergera les futures sauvegardes

Le chemin doit avoir été déclaré dans le fichier **elasticsearch.yml**

```
path.repo: [ "/mnt/nfs/backups", "/media/other_backups" ]
```

SAUVEGARDE : ÉTAPE 2/3

Création d'un **dépôt** (conteneur de sauvegardes)

```
PUT /_snapshot/depot_1
{
  "type": "fs",
  "settings": {
    "location": "/mnt/nfs/backups/depot_1",
    "compress": true
  }
}
```

SAUVEGARDE : ÉTAPE 3/3

Création d'une **sauvegarde** (appelée « snapshot »)

```
PUT  /_snapshot/depot_1/save_1?wait_for_completion=true
{
  "indices": "index_1,index_2"
}
```

Manipulation des sauvegardes

```
GET  /_snapshot/depot_1/_all
DELETE /_snapshot/depot_1/save_1
```

RESTAURATION DES SAUVEGARDES

Enfin, vient l'opération de **restauration**

- Le traitement est long et doit être monitoré
- La sauvegarde peut être restaurée sur un autre cluster

```
POST /_snapshot/depot_1/save_1/_restore
{
  "indices": "index_1,index_2",
  "rename_pattern": "index_(.+)",
  "rename_replacement": "restored_index_$1"
}
```

A l'aide d'une **expression régulière** on peut renommer nos indices "restored_index_1" et "restored_index_2"

SÉCURITÉ

La sécurité est aujourd'hui intégrée à la licence basic

- Chiffrement avec **TLS**
- Accès aux données basé sur des **rôles**
- **Authentication** via mot de passe
- **Espaces de travail** Kibana
- Gestion des droits depuis Kibana
- Gestion de **clés d'API**

CRÉATION DES COMPTES RÉSERVÉS

Première étape : activer la sécurité dans **elasticsearch.yml**

```
xpack.security.enabled: true
```

Lancer la commande de création des comptes/mots de passe

```
$ bin/elasticsearch-setup-passwords interactive
...
Enter password for [elastic]:
Enter password for [apm_system]:
Enter password for [kibana_system]:
Enter password for [logstash_system]:
Enter password for [beats_system]:
Enter password for [remote_monitoring_user]:
```


LES COMPTES RÉSERVÉS

- **elastic** : Le superuser
- **kibana_system** : L'utilisateur avec lequel Kibana est connecté
- **logstash_system** : L'utilisateur avec lequel logstash stocke les informations de monitoring
- **beats_system** : L'utilisateur avec lequel les beats stockent les informations de monitoring
- **apm_system** : L'utilisateur avec lequel le serveur APM stocke les informations de monitoring
- **remote_monitoring_user** : Utilisateur avec lequel le beats MetricBeat stocke ses informations de monitoring

CRÉATION DE RÔLES

Possibilité de créer des rôles avec des **droits restreints** sur les données

```
POST /_security/role/all_privileges_apache_logs
{
  "indices" : [
    {
      "names" : [ "logs-apache-*" ],
      "privileges" : [ "all" ]
    }
  ]
}
```

LES PRIVILÈGES

2 niveaux :

- Cluster
 - Autorisations sur les settings
 - Autorisations sur les templates
 - ...
- Index
 - Autorisations de lecture
 - Autorisations d'écriture
 - Autorisations de création/suppression
 - ...

<https://www.elastic.co/guide/en/elasticsearch/reference/current/security-privileges.html>

CRÉATION DE USER

Avec affectation à un ou plusieurs rôles

```
POST /_security/user/alice
{
  "password" : "myPassword",
  "full_name" : "Alice",
  "email" : "alice@domain.com",
  "roles" : [ "all_privileges_apache_logs" ]
}
```

ILM - GESTION DE CYCLE DE VIE

Elasticsearch offre la possibilité de réaliser des **actions** de manière **automatique** sur les index.

- Lorsque l'index atteint une **certaine taille**, ou un **nombre de documents** donné
- Lorsque l'index existe depuis un temps donné

Cela permet d'automatiser la **politique de rétention** de données choisie (ou imposée)

CYCLE DE VIE

4 phases principales :

- **Hot**: L'index est actif, on y injecte des données, et on les exploite.
- **Warm**: L'index est en lecture seule, mais toujours exploité.
- **Cold**: L'index est en lecture seule, et faiblement exploité. Les requêtes peuvent être plus lente, ce n'est pas un problème.
- **Delete**: L'index n'est plus nécessaire, peut (doit?) être effacé.

ACTIONS EN FONCTION DE LA PHASE

Elasticsearch offre la possibilité d'effectuer **une ou plusieurs actions** pour une phase donnée.

Cela peut être automatisé

<https://www.elastic.co/guide/en/elasticsearch/reference/current/ilm-index-lifecycle.html>

PHASE HOT

Support des actions suivantes :

- Set Priority (phase de recover plus rapide que les autres index)
- Unfollow (gestion de la réplication cross cluster)
- Rollover (explosion d'un index en multiple index)

PHASE WARM

Support des actions suivantes :

- Set Priority
- Unfollow
- Read-Only (Passage en lecture seule)
- Allocate (Choix des noeuds pour distribuer les shards)
- Shrink (Rétrécissement de l'index - optimisations)
- Force Merge (optimisations des segments Lucene)

PHASE COLD

Support des actions suivantes :

- Set Priority
- Unfollow
- Allocate
- Freeze (Ferme l'index. Ré-ouvre si requêté.)
- Searchable Snapshot (Crée un snapshot qui peut être requêté)

PHASE DELETE

Support des actions suivantes :

- Searchable Snapshot
- Delete (suppression de l'index et du snapshot)

EXEMPLE DE CONFIGURATION

```
PUT _ilm/policy/ma_regle_1
{
  "policy": {
    "phases": {
      "hot": {
        "actions": { "rollover": { "max_size": "25GB" } }
      },
      "delete": {
        "min_age": "30d",
        "actions": { "delete": {} }
      }
    }
  }
}
```

Explosion en multiples index de 25GB, et suppression au bout de 30 jours

AFFECTATION DE LA POLITIQUE DE RÉTENTION (1-2)

Les politiques de rétention sont affectées aux templates

```
PUT _template/mon_template
{
  "index_patterns": ["logs-*"],
  "settings": {
    "index.lifecycle.name": "ma_regle_1"
  }
}
```

Dès qu'un nouvel index respectant le pattern est créé, la politique lui est affectée.

AFFECTATION DE LA POLITIQUE DE RÉTENTION (2-2)

Possibilité d'affecter une politique de rétention à un index existant

```
PUT logs-apache-2025-06/_settings
{
  "index": {
    "lifecycle": {
      "name": "ma_regle_1"
    }
  }
}
```

FIN DE LA FORMATION

Merci pour votre participation

GLOSSAIRE

- **Analyse** : processus de conversion texte => tableau de mots
- **Cluster** : regroupement de noeuds partageant les données
- **Document** : Objet JSON représentant la donnée.
- **Field** : Partie d'un document.
- **Filter** : Recherche n'influant pas sur le score.
- **Id** : identifiant unique d'un document

GLOSSAIRE

- **Index** : Equivalent d'une table relationnelle
- **Index template** : définit les mappings et settings à appliquer pour les nouveaux index.
- **Mapping** : définition du schéma dans un index.
- **Node** : Noeud Elasticsearch (instance).
- **Primary shard** : shard primaire, actif (indexation/recherche)

GLOSSAIRE

- **Query** : requête Elasticsearch
- **Recovery** : phase de recouvrement des index, lancé automatiquement au démarrage.
- **Reindex** : re-écriture des données d'un index dans un nouvel index.
- **Replica shard** : copie d'un shard permettant la disponibilité en cas de défaillance d'un noeud qui contient un shard primaire.

GLOSSAIRE

- **Routing** : route logique d'un document vers un shard.
- **Shard** : partition d'index, instance Lucene.

LIENS UTILES

Référence Elasticsearch

<https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html>

Référence Kibana

<https://www.elastic.co/guide/en/kibana/current/index.html>

Référence Logstash

<https://www.elastic.co/guide/en/logstash/current/index.html>

LIENS UTILES

Dépôts github

- <https://github.com/elastic/elasticsearch>
- <https://github.com/elastic/kibana>
- <https://github.com/elastic/logstash>
- <https://github.com/elastic/elasticsearch-py>
- <https://github.com/elastic/elasticsearch-php>

Docker

- https://hub.docker.com/_/elasticsearch
- https://hub.docker.com/_/kibana

DOCDOKU

Retrouvez nous sur <https://www.docdoku.com>

Twitter

<https://twitter.com/docdoku>

Facebook

<https://www.facebook.com/docdoku>

Linkedin

<https://fr.linkedin.com/company/docdoku>

VOTRE FORMATEUR

Eric Descargues

eric.descargues@docdoku.com

06 70 00 12 91



DocDoku a pour mission d'aider les équipes et entreprises tech à **créer, déployer** et **maintenir** leurs **plateformes data** et **cloud** : <https://www.docdoku.com/>