Elasticsearch

Semifir

Enzo Radnaï / Adrien Vossough

Introduction

Qu'est-ce qu'Elasticsearch ?

- Projet open source développé en Java dont la 1ère version est sortie en 2010
- Basé sur la librairie Apache Lucene et les acquis du projet Compass
- Serveur de moteur de recherche capable de fonctionner en cluster
  - API REST/JSON
  - Recherche distribuée en mode cluster
  - Taillé pour de gros volumes de données
  - Configuration en JSON et YAML
  - Structure de données non figée
  - Recherche multi-index
  - Support des facettes, géolocalisation

Qu'est-ce que Solr ?

- Projet open source développé en Java dont la 1ère version est sortie en 2006
- Basé sur Apache Lucene
- Fusionné avec Lucene en 2011
- Application Web (WAR) fournissant un serveur de moteur de recherche
  - API HTTP/XML, JSON
  - Recherche distribuée
  - Interface d'administration
  - Réplication maître/esclave
  - Support des facettes, géolocalisation
  - Outils d'import de données (CSV, XML, Base de données)
  - Configuration par fichiers XML

Entre le projet Solr et Elasticsearch :

- Sorti plus tard que Solr, Elasticsearch est considéré comme plus intuitif dans sa gestion des clusters
- Solr utilise ZooKeeper pour la coordination des clusters alors qu'Elasticsearch a son propre système interne
- Les deux peuvent partitionner les index mais Solr peut le faire dynamiquement sans tout réindexer contrairement à Elasticsearch
- Elasticsearch gère autoamtiquement la répartition de charges entre les différentes partitions lors d'ajout de machines à chaud contrairement à Solr
- Les deux permettent l'indexation de nouveaux champs des documents à chaud
- Elasticsearch fabrique automatiquement les schémas des documents
- Elasticsearch autorise l'imbrication de types (liste dans les listes dans les objets, etc.)

Points forts

Vitesse : Des recherches quasiment en temps réel.

Scalabilité : D'un serveur local à une architecture distribuée gérant des péta octets de données

Résilience : Haute disponibilité, réplication et sauvegarde automatique

Modulaire : Un système de plugins permet d'ajouter des fonctionnalités à celles de base proposées par Elasticsearch

- sécurité
- monitoring
- alerting
- reporting

Simplicité : Elasticsearch dissimule toute la complexité derrière l'API. Il est possible de l'utiliser avec la configuration par défaut, ou de tout redéfinir manuellement : indexation, réplication, clusterisation, tout se fait automatiquement.

Architecture

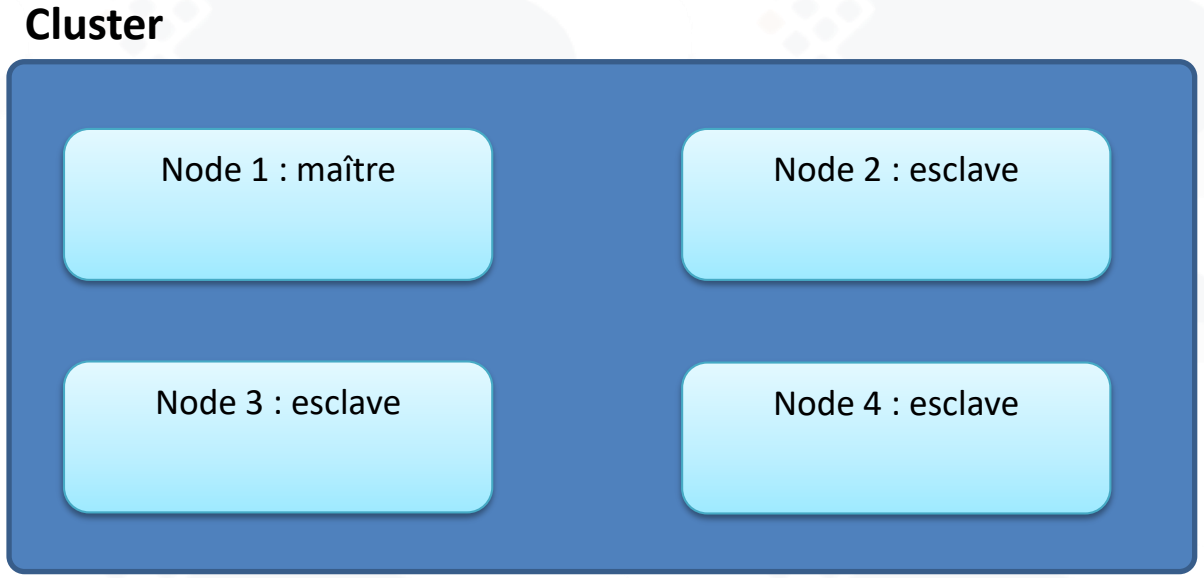
Introduction

- On accède aux données au travers de requêtes HTTP (PUT, GET, POST ou DELETE).
- Elasticsearch est **Near-Real-Time** (NRT), les données ajoutées sont rapidement exploitables
- Elasticsearch est composé d'un ensemble de NODES, chacun étant sur un serveur Les ensembles de nodes sont regroupés en CLUSTER.
- Le cluster est la représentation d'une ou plusieurs bases de données. Toutes les données sont réparties sur les différents nodes.
- Tous les nodes présents dans le cluster sont conscients de l'existence des autres nodes du même cluster. A ce titre, chaque node peut, par défaut, recevoir les requêtes HTTP, les traiter ou les transmettre aux autres nodes.

Clusters et nœuds

Nœud :
Correspond à un processus/instance d'Elasticsearch en cours d'exécution.
Ils peuvent être placés sur plusieurs machines

Cluster :
Un cluster est composé d'un à plusieurs nœuds. Un nœud maître est choisi, il sera remplacé en cas de défaillance



L'intérêt d'avoir plusieurs instances/nœuds Elasticsearch permet d'augmenter les performances et la disponibilité du moteur de recherche.

Clusters et nœuds

- Par défaut chaque nœud du cluster peut être le MASTER NODE, c'est-à-dire le node qui gère le cluster.
- Chaque node et chaque cluster est représenté par un identifiant unique.
- Par défaut, il existe un cluster dont l'identifiant est Elasticsearch.
- Les nodes sont quant à eux représentés par un UUID et sont associés par défaut au cluster Elasticsearch.
- Il est possible d'attacher les nodes à un autre cluster lors de leur création.

Un node peut avoir quatre rôles possibles:

- **master eligible node** : peut être le master node
- **data node** : peut contenir des données
- **ingest node** : peut recevoir des requêtes
- **coordinating node only** : ne peut que recevoir les requêtes

Seuls les trois premiers rôles peuvent être actif simultanément

Index et shards

Index

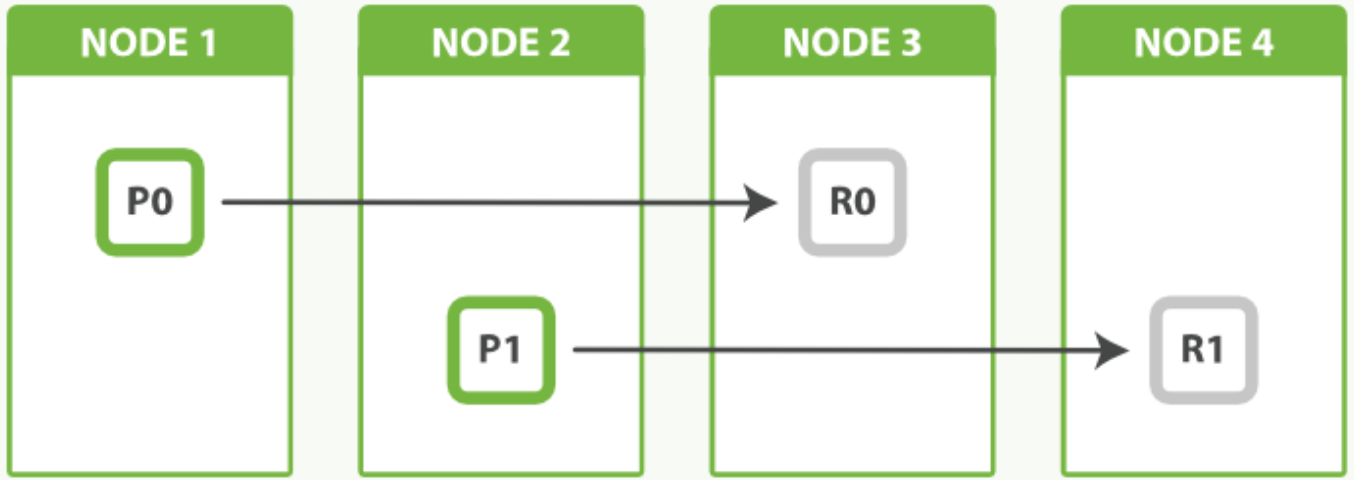
Un index est un espace logique de stockage de documents de même type. Il a un identifiant unique et est découpé sur un à plusieurs Primary Shards. Un index peut être répliqué sur zéro ou plusieurs Secondary Shards.

Primary Shards

C'est une partition de l'index. Par défaut, l'index est découpé en 5 Shards Primary. Il n'est pas possible de changer le nombre de Shards après sa création.

Secondary Shards ou Replica

Il s'agit de de copies de Primary Shards. Il peut y en avoir zéro à plusieurs par Primary Shard. Ce comportement est adopté à des fin de performance et de sécurisation des données.



Px : primary shards
Rx : replicat

Index et shards

Plus de shards améliore les performances à l'indexation et permet de distribuer un index volumineux sur plusieurs nœuds

Plus de réplicas améliore les performances à la recherche et la disponibilité de l'index

Document

Un document est un simple enregistrement immutable dans un shard Elasticsearch. Un document est structuré comme un objet JSON et doit appartenir à un type (qui définit sa structure).

Un document de type livre

```
{
    titre :  Eloquent JavaScript, Second Edition ,
    auteur :  Marijn Haverbeke ,
    published :  2014-12-14T00:00:00.000Z ,
    editeur :  No Starch Press ,
    pages : 472,
    description :  JavaScript lies at the heart of …
}
```

Un document de type ville

```
{
    nom :  Lille ,
    code_postal :  59000 ,
    url :  https://lille.fr
}
```

Les documents sont enregistrés et assignés par type.

En résumé

Représentation de la structure logique :

Index

Type : ville

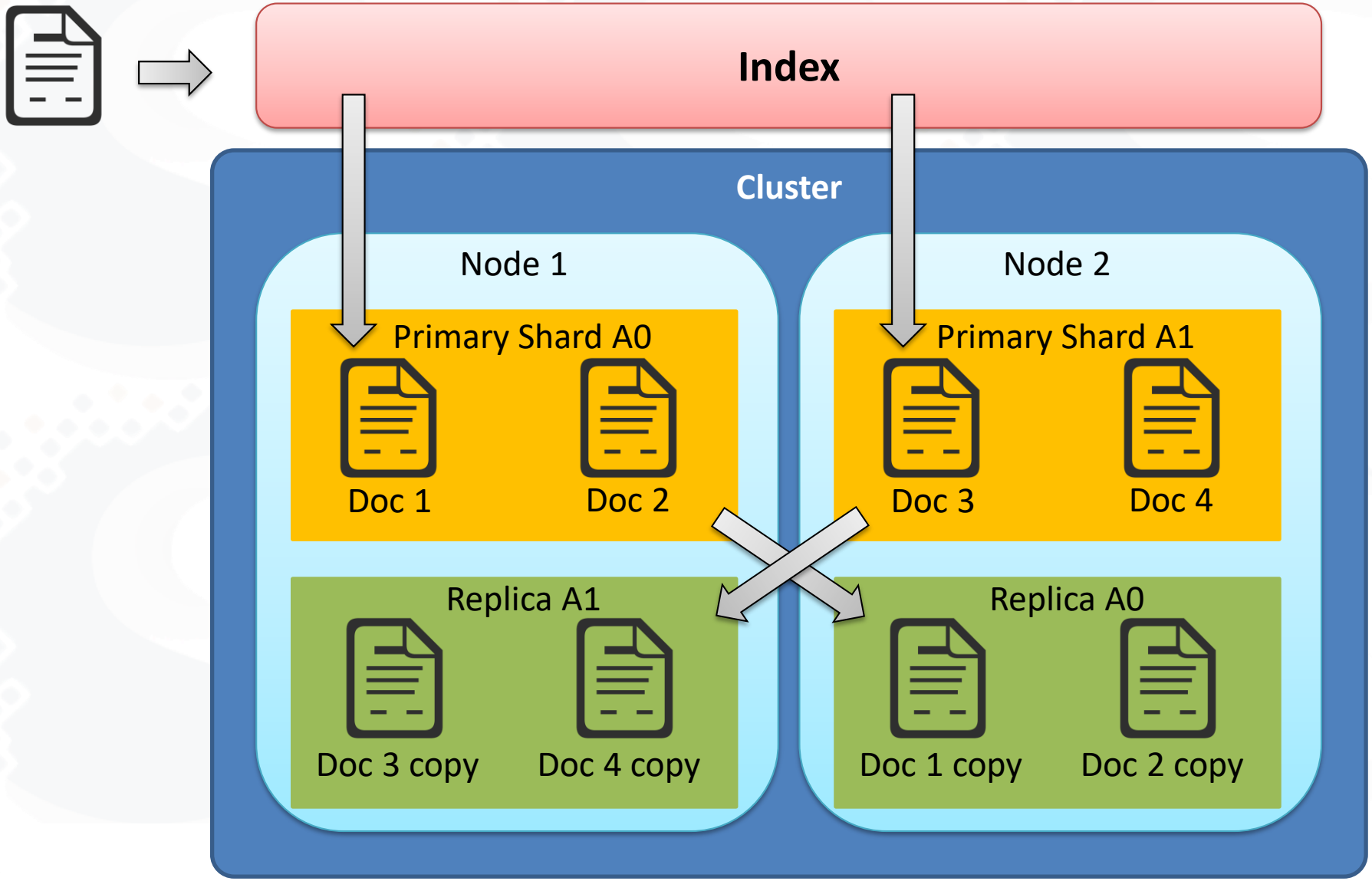
id	nom	code_postal	url
1	Lille	59000	https://lille.fr
2	Paris	75000	https://paris.fr

Type : livre

id	auteur	titre	editeur
4	Marijn Haverbeke	Eloquent JavaScript	No Starch Press
5	Adrien Vossough	Cours ES	Semifir

Réplication des documents

Si un nœud vient à s'arrêter, l'ensemble des documents est répliqué ce qui assure le bon fonctionnement d'ES



En résumé

Si nous devions simplifier :

SGBDR	Elasticsearch
Base de donnée	Index
Table	Type
Ligne	Document
Colonne	Champ ou propriété

Attention : ceci est une analogie, mais n'est pas vrai dans le mode de fonctionnement. Dans Elasticsearch les types sont fortement liées.

En résumé :
Un cluster Elasticsearch peut contenir plusieurs **index** (bases de données) qui, à leur tour, contiennent plusieurs **types** (tables). Ces types contiennent plusieurs **documents** (lignes) et chaque document possède des propriétés (colonnes).

Remarque

Les types sont ***deprecated*** et sont voués à disparaître dans les versions futurs d'Elasticsearch. Il est maintenant conseillé de créer un index par type de document.

Jusqu'à la version 6.2, il faut mettre un type dans les URI des requêtes ce qui donnerait par exemple:
PUT /nom_index/nom_type/id

Depuis la version 6.2, le type est remplacé par _doc soit PUT /index/_doc/id

Par conséquent, si l'on travaille avec une version antérieure à la 6.2, il est recommandé d'utiliser le mot default (PUT /nom_index/default/id) partout à la place du type afin d'avoir une pratique s'approchant du comportement des versions actuelles.

Les types disparaissent définitivement à la version 8 d'Elasticsearch.

https://www.elastic.co/guide/en/elasticsearch/reference/6.7/removal-of-types.html

Réplication des documents

Le cluster est un ensemble de documents regroupés en INDICES (aka bases de données) avec un identifiant unique.

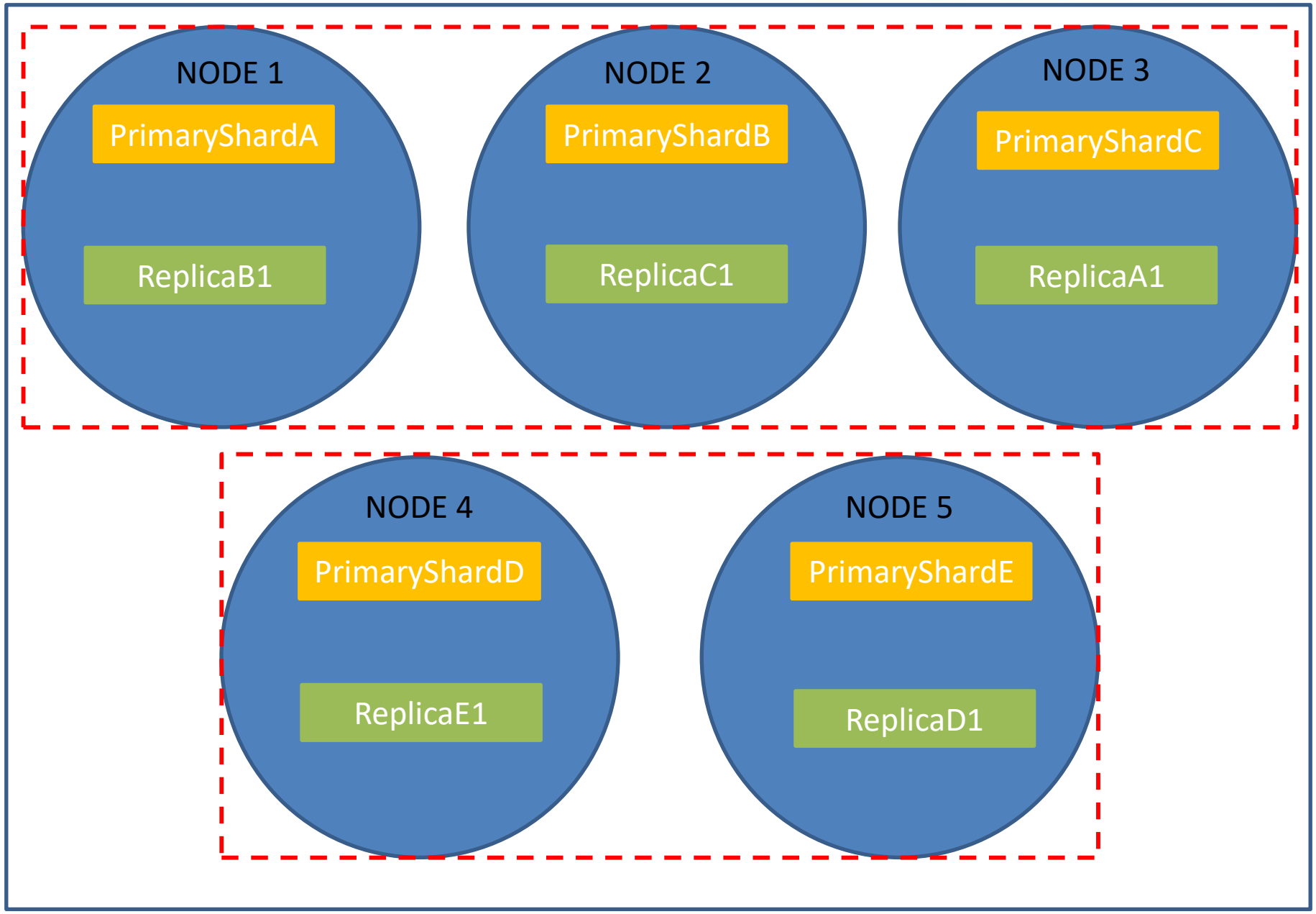
Un indice est un ensemble d'INDEX.

Un index pourrait être considéré comme une table de la base de donnée. Le nom de l'index est unique et défini lors sa création. C'est lors de la création de l'index qu'est déterminé le nombre de shards qui seront utilisés par celui-ci.

Un document se voit attribué un identifiant automatiquement par Elasticsearch à moins qu'il soit spécifié explicitement lors de son ajout.

C'est au travers de ces différents noms et identifiants uniques qu'Elasticsearch peut retrouver les documents.

Représentation Indice



Indice

Index

Fonctionnement

Sharding

Chaque node est une instance d'Elasticsearch pouvant être lancé sur différent serveurs.

Le sharding est la répartition des données entres les nodes d'un cluster.

Par exemple, avec 1To de données mais que le cluster qui doit le contenir est composé de deux nodes sur des serveurs de 500go chacun. Le sharding permettra de séparer notre index en deux.

L'intérêt avec le sharding:

- Répartir nos données dans le cluster.
- Permettre à plusieurs nodes d'exécuter la requête en même temps ce qui donne un traitement plus rapide.

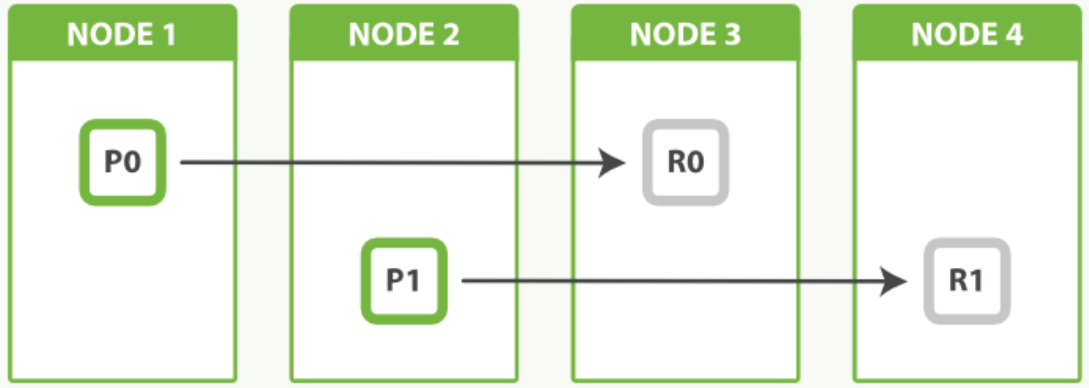
Sharding

Le nombre de shards est défini lors de la création ; par défaut il est de 5.

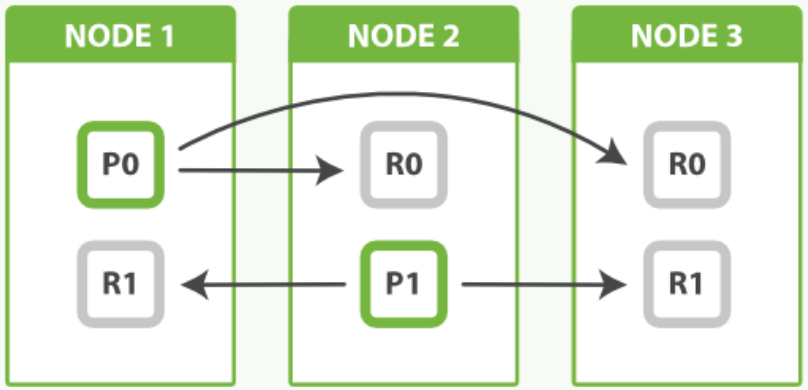
Elasticsearch ne permet pas de modifier cette valeur après la création de l'index, il faudra dans ce cas le reconstruire.

Les shards sont ensuite clonés, ceux-ci sont appelé Replica ou shard secondaire . Cela empêche la perte d'information en cas de panne d'un node et permet d'accélérer le traitement des requêtes de recherche.

Par défaut seul un réplica est créé par shard, cette valeur peut être changé.



1 réplica par shard



2 réplicas par shard

Sharding

Pour synchroniser l'ensemble des informations, les écritures passent toujours par le shard puis transmises aux réplicas.

Routing

Le routing permet de placer le document dans le bon shard.

Pour se faire, on utilise une formule très simple:

- shard_num = hash(ID_doc) % num_primary_shards

L'ID du document passe par une fonction de hashage ce qui ressort un nombre entier sur lequel ont fait un modulo avec le nombre total de primary shard.
Le résultat donne le numéro du shard dans lequel on doit ranger le document.

Ce calcul permet de déduire que :
Nous ne pouvons pas modifier le nombre de shard au risque de perdre des documents .

Requêtes

Les requêtes pour les recherches sont au format **Query DSL** (Domain Specific Language) basé sur le JSON.

Lors de l'envoi d'une requête au cluster :

1. La requête est transmise à un node disponible.
2. Le node devient le ***coordinating node*** avec le point d'entrée de la requête puis la communique aux autres nodes de l'index correspondant.
3. Le coordinating node récupère le résultat et le renvoi.

Si la requête contient l'ID du document, seul le shard en question est interrogé car elle est envoyé directement au bon endroit.

Par défaut, tous les nodes peuvent être un coordinating node.

Requêtes

Les requêtes pour les recherches sont au format **Query DSL** (Domain Specific Language) basé sur le JSON.

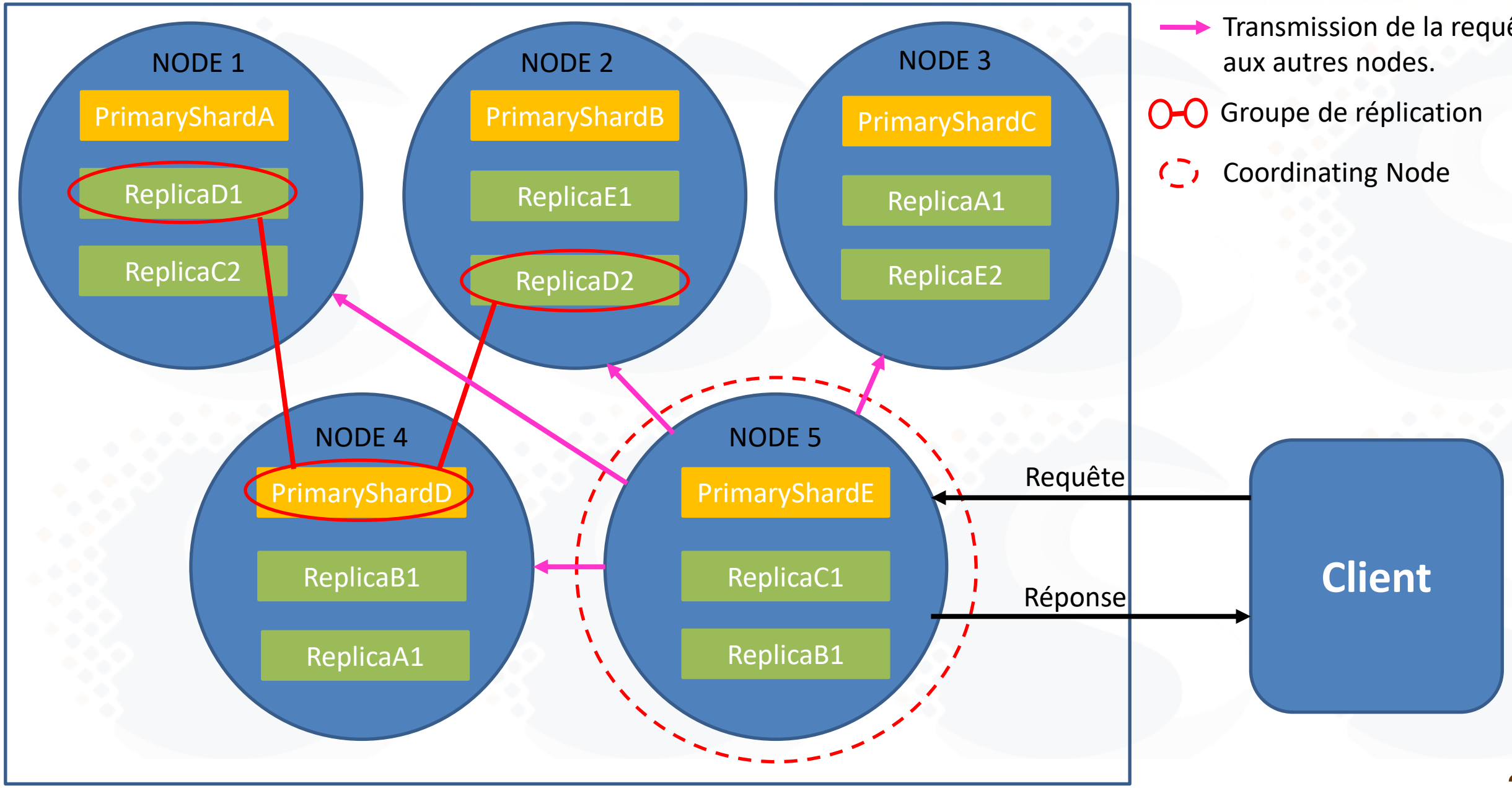
Lors de l'envoi d'une requête au cluster :

1. La requête est transmise à un node disponible.
2. Le node devient le ***coordinating node*** avec le point d'entrée de la requête puis la communique aux autres nodes de l'index correspondant.
3. Le coordinating node récupère le résultat et le renvoi.

Si la requête contient l'ID du document, seul le shard en question est interrogé car elle est envoyé directement au bon endroit.

Par défaut, tous les nodes peuvent être un coordinating node.

Requêtes


NODE 1
PrimaryShardA
ReplicaD1
ReplicaC2
NODE 2
PrimaryShardB
ReplicaE1
ReplicaD2
NODE 3
PrimaryShardC
ReplicaA1
ReplicaE2
NODE 4
PrimaryShardD
ReplicaB1
ReplicaA1
NODE 5
PrimaryShardE
ReplicaC1
ReplicaB1
Transmission de la requête aux autres nodes.
Groupe de réplication
Coordinating Node
Requête
Réponse
Client