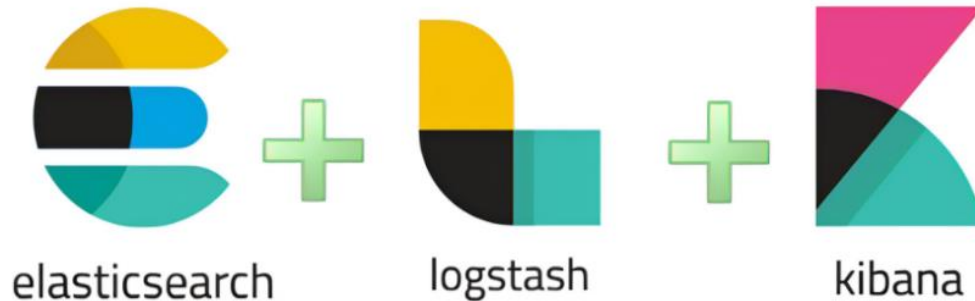


## Chapitre3 ElasticSearch

**Mr DIATTARA Ibrahima**



# Sommaire

1. Elasticsearch
2. Xpack
3. Kibana
4. Logstash
5. Architecture ELK

# Elasticsearch

Elasticsearch est un **moteur de recherche** et d'analyse distribué en temps réel, Il est utilisé pour:

- Recherche full text
- Recherche structurée
- L'analyse

<b>Relational DB</b>	Base de données	Tables	Lignes	Colonnes
<b>Mongo DB</b>	Base de données	Collections	Documents	Champs
<b>Elasticsearch</b>	Index	Types	Documents	Champs

Un Document possède un type (qui défini son mapping) et chaque document est relié par un id

# Démarrage

Deux ports par défaut:

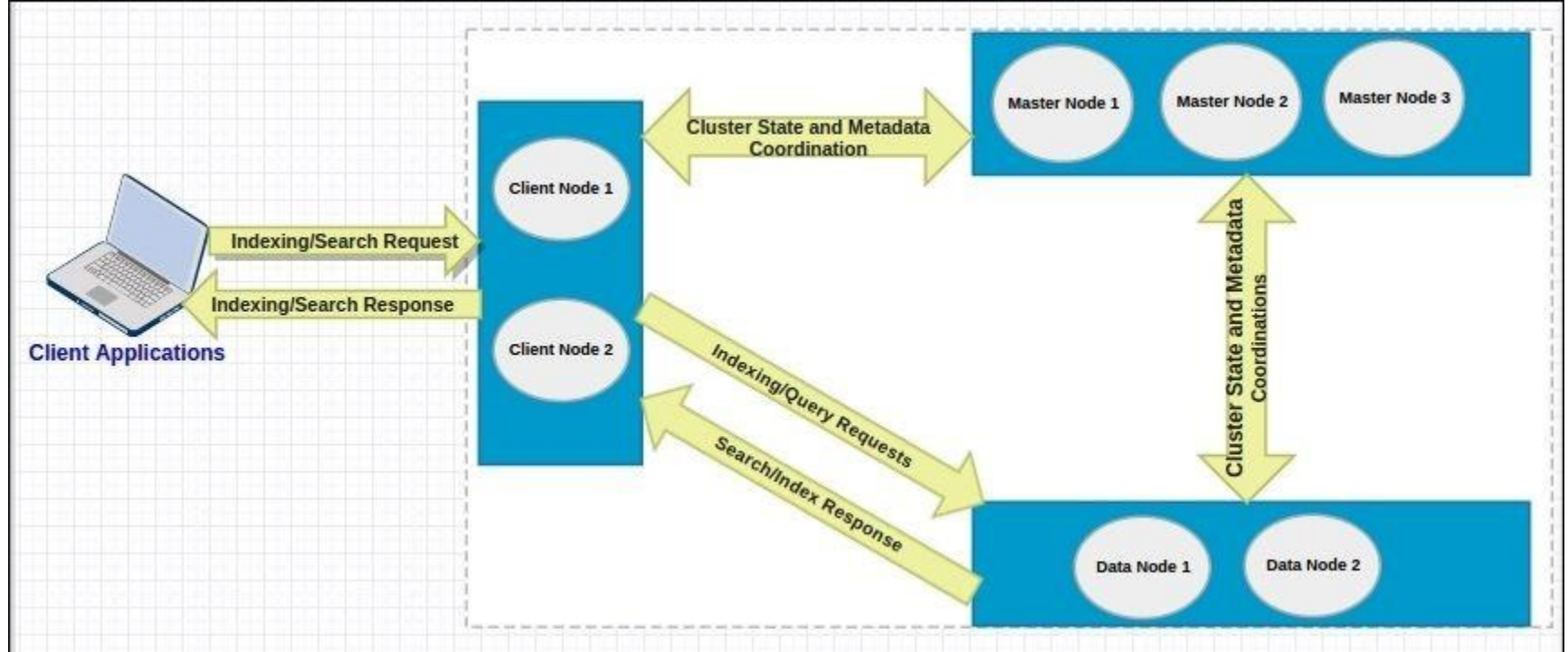
- **9200** ⇒ http (requête / ingestion)
- **9300** ⇒ transport (inter-node communications)

← → ↻ ⚠ Not secure | http://mosef02.westeurope.cloudapp.azure.com:9200

```
{
  "name" : "elk-mosef1",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "HPfHb2S2RZ6YBjwZtX1NoQ",
  "version" : {
    "number" : "7.16.3",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "4e6e4eab2297e949ec994e688dad46290d018022",
    "build_date" : "2022-01-06T23:43:02.825887787Z",
    "build_snapshot" : false,
    "lucene_version" : "8.10.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

<http://mosef02.westeurope.cloudapp.azure.com:9200>

# Types des noeuds Elasticsearch



# Types des noeuds Elasticsearch

## Master Node

- responsable des actions légères à l'échelle du cluster, telles que la création ou la suppression d'un index, le suivi des nœuds faisant partie du cluster et le choix des shards à allouer à quels nœuds.
- Ne stocke pas de données
- Config : (elasticsearch.yml)
  - Elasticsearch version 7
    - `node.roles: [ master ]`

## Data Node

- Les nœuds "data" effectuent des opérations liées aux données telles que CRUD, recherche et agrégations.
- stocke les données
- Config: (elasticsearch.yml)
  - Elasticsearch version 7
    - `node.roles: [ data ]`

# Types des noeuds Elasticsearch

## Client Node (coordinating node)

- Routage des requetes et load balancer
- Ne stocke pas de données
- Config : (elasticsearch.yml)
  - Elasticsearch version 7
    - node.roles: [ ]

PS : d'autres rôles ont été ajouté récemment dans les dernières versions d'Elasticsearch , tels que : ml , ingest ...

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

# Exemple

Comment lister les noeuds du cluster Elasticsearch?

⇒ l'API `/_cat/nodes?v`

← → ↻ ⚠ Not secure | http://mosef02.westeurope.cloudapp.azure.com:9200/\_cat/nodes?v

ip	heap.percent	ram.percent	cpu	load_1m	load_5m	load_15m	node.role	master	name
10.0.0.9	34	97	2	0.03	0.09	0.04	cdfhilmrstw	*	elk-mosef1

← → ↻ ⚠ Non sécurisé | ec2-35-181-53-245.eu-west-3.compute.amazonaws.com:9200/\_cat/nodes?v

ip	heap.percent	ram.percent	cpu	load_1m	load_5m	load_15m	node.role	master	name
172.31.35.98	17	71	1	0.93	0.32	0.11	d	-	node-2
172.31.45.249	21	33	0	0.07	0.04	0.05	dm	*	node-1



# Index Elasticsearch

Les **documents JSON** sont stockés dans un ou plusieurs **index** Elasticsearch.

Un **index** est l'équivalent d'**une table** SQL.

Chaque index peut être raffiné avec la notion de **type**, correspondant à une sous-catégorie de l'index que l'on pourra spécifier au besoin. Tous les types d'un index partagent le même **schéma** de documents JSON.

# Les Shards

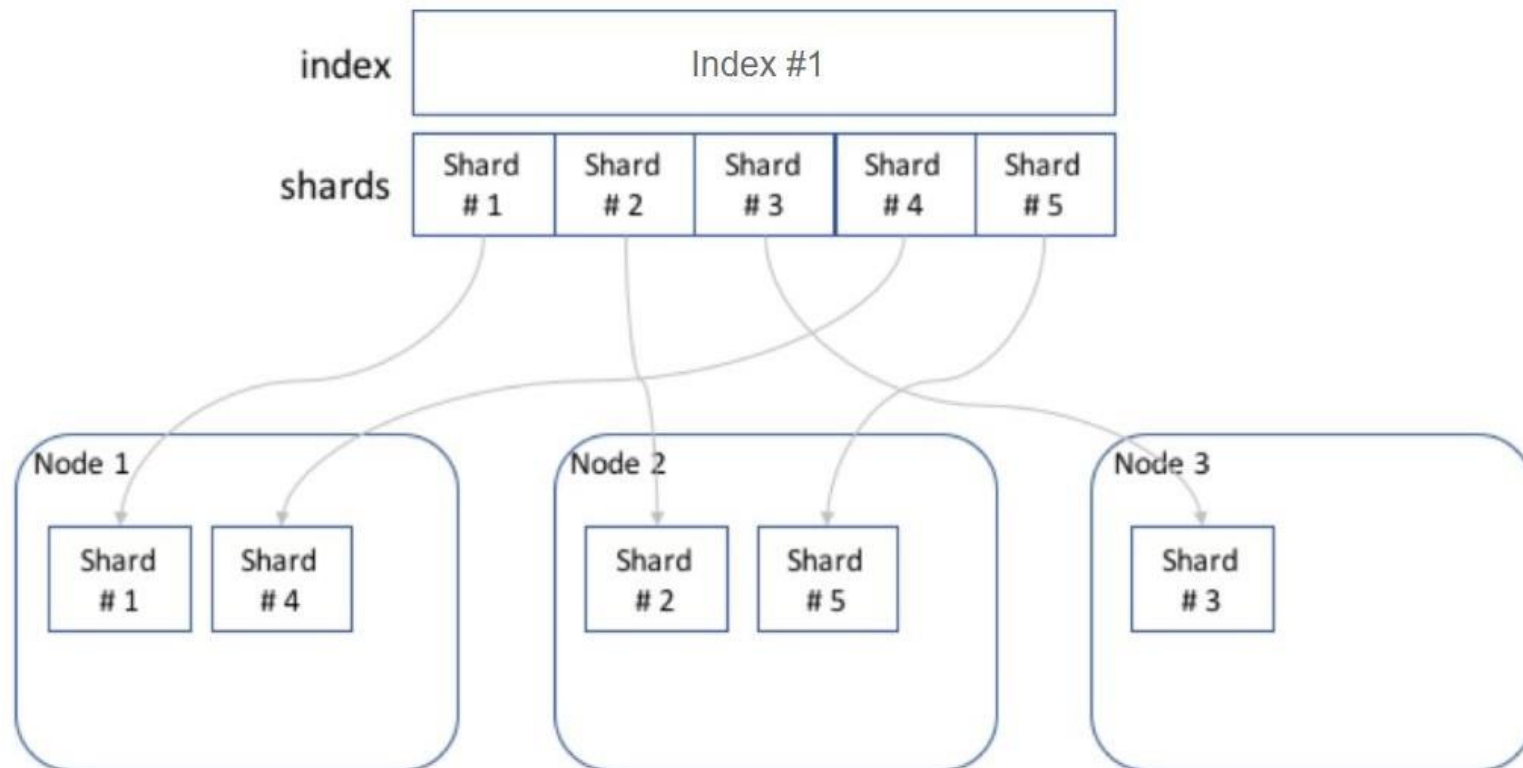
Un index Elasticsearch est **partitionné en un ou plusieurs shards**

2 types de shards:

- **Primary** : shard primaire
- **Replicat** : shard répliat (une copie du shard primaire)

Le fait d'avoir des shards “replicat” nous permet d'avoir un cluster Elasticsearch **résilient aux pannes**

# Index / Shard



# Définition

---

## ❑ Index

- Un peu comme une base de données sur un SGBDR relationnel
- Une collection de document qui ont tous un points commun (

## ❑ Type Mapping

- Le mapping est similaire au schéma du type
- Le mapping peut être défini manuellement, mais aussi généré automatiquement quand les documents sont indexés

## ❑ Shard

- Découper un index en plusieurs parties pour y distribuer les documents
- C'est l'équivalent des partitions dans un SGBDR
- Nos documents sont stockés et indexés dans les Shards, mais nous ne nous adressons pas directement à eux : nos applications s'adressent à un index

## ❑ Réplica

- Recopie d'un shard en une ou plusieurs copie dans l'ensemble du cluster
- Un Shard replica : est une copie d'un Shard primaire (similaire au RAID 1)

## ❑ Alias

- C'est l'équivalent d'une vue dans le monde SGBDR
- Un alias ElasticSearch peut être configuré de manière à pointer vers un ou plusieurs indexes d'un cluster tout en spécifiant des filtres ou des clés de routage

# Exemple

Comment lister les index Elasticsearch dans l'ordre décroissant ?

⇒ l'API `/_cat/indices?v`

[http://mosef02.westeurope.cloudapp.azure.com:9200/\\_cat/indices?v=true&s=store.size:desc](http://mosef02.westeurope.cloudapp.azure.com:9200/_cat/indices?v=true&s=store.size:desc)

← → ↻ ⚠ Not secure   http://mosef02.westeurope.cloudapp.azure.com:9200/_cat/indices?v=true&s=store.size:desc									
health	status	index	uuid	pri	rep	docs.count	docs.deleted	store.size	pri.store.size
yellow	open	maas_cheikhyakhoub_one_shot	kE4c5IxuT6yTnz69g7V2bQ	1	1	32000	96000	42.9mb	42.9mb
green	open	.geopip_databases	Tn5GovQoTdamj-PAF2RzUA	1	0	41	38	38.9mb	38.9mb
yellow	open	ibrahima_camara_streaming	Zn04GwGrRjue9IdTQkCfDA	2	1	77184	0	25mb	25mb
yellow	open	eyabenalaya_oneshot	p06qSwizQbqD9D6jBbGgXg	3	2	32000	9500	14.8mb	14.8mb
green	open	khadija_projet_kibana	EPNGCGlCS1Cm0xJ4QmkmqQ	1	0	32000	7276	13.8mb	13.8mb
yellow	open	kane_oumar_streaming	vil7QWfoT1KRDWhNENrTLw	3	1	32001	0	12.8mb	12.8mb
yellow	open	kane_oumar_one_shot	VD81re3QQ0esfKmbAluANQ	3	2	32000	0	12.5mb	12.5mb
yellow	open	youssouphie_one_shot	LdaWjtnCT56CjBzCMyyxSg	3	2	30070	3284	12.2mb	12.2mb
yellow	open	papabagaye_streaming	-ELxF93pSYy6-cDvCRoMiA	3	2	32022	0	12.2mb	12.2mb
yellow	open	ngomstreaming	pnHtCM4d75aXaMAsePkLlg	3	2	32001	0	12mb	12mb
yellow	open	samsidine_projet_kiban	S5pkXw0lQBmIQqtAtMe-9g	3	2	32000	0	12mb	12mb
yellow	open	papabagaye_one-shot	Nq1BYJ-7TTK4fZ8f44-PEg	3	2	32000	0	12mb	12mb
yellow	open	ngomoneshot	MLX0RA0kTEys7fXwyKAagg	3	2	32000	0	11.9mb	11.9mb
yellow	open	ibrahima_camara_projet_kibana	OJ3BktwJTUa8NbR6weLw0w	2	1	32000	0	11.9mb	11.9mb
yellow	open	eyabenalaya_index	8d1gbxehQxGeBcmXn8qlVg	1	1	32000	0	11.6mb	11.6mb
green	open	abdoukarim_projet_kibana	C80diXTmQFmK8KVOKIpbA	1	0	32000	0	11.2mb	11.2mb
yellow	open	moustapha_ndiaye_one-shot	cBZF6nLNRcOA24E5_L7bdA	3	2	32000	0	11.1mb	11.1mb
yellow	open	maas_cheikhyakhoub_streaming	JrV5FFtAT2-Zb8PG-890xQ	1	1	32004	0	11.1mb	11.1mb
yellow	open	papasambadia_streaming	wo2XAXA2Qf094m0ToCY8zA	1	1	32001	0	11.1mb	11.1mb
yellow	open	khadiim_mbacke_ndiaye_oneshoot	HR70Bv2aTG55Ag1W744bgw	1	1	32000	0	11.1mb	11.1mb
yellow	open	papasambadia_projet_kibana	mkJYyU8UTE-8A1NKI450vw	1	1	32000	0	11.1mb	11.1mb
yellow	open	khalifa	HjxV1hTwSu-yWIf1g15f0Q	3	1	30072	0	10.9mb	10.9mb

# Exemple

Comment lister les shards Elasticsearch ?

⇒ l'API `/_cat/shards?v`

← → ↻ ⚠ Non sécurisé   ec2-15-236-206-197.eu-west-3.compute.amazonaws.com:9200/_cat/shards?v							
index	shard	prirep	state	docs	store	ip	node
kibana_sample_data_flights	0	p	STARTED	13059	5.5mb	172.31.45.249	ip-172-31-45-249.eu-west-3.compute.internal
ilm-history-3-000001	0	p	STARTED			172.31.45.249	ip-172-31-45-249.eu-west-3.compute.internal
.kibana-event-log-7.10.0-000001	0	p	STARTED	6	27.5kb	172.31.45.249	ip-172-31-45-249.eu-west-3.compute.internal
.kibana_1	0	p	STARTED	95	10.4mb	172.31.45.249	ip-172-31-45-249.eu-west-3.compute.internal
.apm-custom-link	0	p	STARTED	0	208b	172.31.45.249	ip-172-31-45-249.eu-west-3.compute.internal
.async-search	0	p	STARTED	0	231b	172.31.45.249	ip-172-31-45-249.eu-west-3.compute.internal
.kibana_task_manager_1	0	p	STARTED	5	120.5kb	172.31.45.249	ip-172-31-45-249.eu-west-3.compute.internal
.apm-agent-configuration	0	p	STARTED	0	208b	172.31.45.249	ip-172-31-45-249.eu-west-3.compute.internal

# Comment vérifier l'état du cluster Elasticsearch ?

⇒ l'API `/_cluster/health?pretty`

← → ↻ ⚠ Non sécurisé | ec2-35-181-53-245.eu-west-3.compute.amazonaws.com:9200/\_cluster/health?pretty

```
{
  "cluster_name" : "formation_big_data",
  "status" : "green",
  "timed_out" : false,
  "number_of_nodes" : 2,
  "number_of_data_nodes" : 2,
  "active_primary_shards" : 10,
  "active_shards" : 20,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 0,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 100.0
}
```

← → ↻ ⚠ Non sécurisé | 20.101.123.129:8082/\_cluster/health?pretty

```
{
  "cluster_name" : "elasticsearch",
  "status" : "yellow",
  "timed_out" : false,
  "number_of_nodes" : 1,
  "number_of_data_nodes" : 1,
  "active_primary_shards" : 50,
  "active_shards" : 50,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 38,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 56.81818181818182
}
```

# Status du cluster Elasticsearch

- **Green** : tous les shards primaires et répliqués sont assignés à des data nodes
- **Yellow** : un ou plusieurs shards répliqués sont non assignés
- **Red** : un ou plusieurs shards primaires sont non assignés



Indexation

# Exercice1

1 Coder un script python qui permet de stocker le document JSON suivant dans un index "prenom\_chiffre" (exemple : ibrahima\_1)

Document JSON:

```
{  
  "location": "14.76, -14.76",  
  "typeproduit": "electronique",  
  "prix": 220,  
  "agent_timestamp": datetime.utcnow().strftime('%Y-%m-%dT%H:%M:%SZ')  
}
```

2 consulter la data

[http://clustersdaelatsic.eastus.cloudapp.azure.com:9200/index\\_name/\\_search?pretty](http://clustersdaelatsic.eastus.cloudapp.azure.com:9200/index_name/_search?pretty)

3 Récupérer le schéma de votre index avec la requête:

[http://clustersdaelatsic.eastus.cloudapp.azure.com:9200/ibra1/\\_mapping?pretty](http://clustersdaelatsic.eastus.cloudapp.azure.com:9200/ibra1/_mapping?pretty)

4 Que constatez-vous pour le type location ? / trouvez une solution pour rendre le type location en geo pointe

## Solution:

[https://github.com/idiattara/Spark\\_DIATTARA/blob/main/post\\_elastic.py](https://github.com/idiattara/Spark_DIATTARA/blob/main/post_elastic.py)

[https://github.com/idiattara/Spark\\_DIATTARA/blob/main/mapping\\_elastic](https://github.com/idiattara/Spark_DIATTARA/blob/main/mapping_elastic)

# Exercice2

1 Coder un script python qui permet d'envoyer le document JSON dans Kafka

```
{
  "location": "14.76, -14.76",
  "typeProduit": "coca",
  "price": 100
}
```

2 Récupérer la data avec NIFI schéma en rajoutant un timestamp de format `${now():format("yyyy-MM-dd'T'HH:mm:ss'Z'", "GMT")}`

3 Créer le mapping de vos données sous elastic

4 Utiliser le processor putElasticHttp pour mettre la data dans elastic

5 Sous kibana créer:

Un discover data

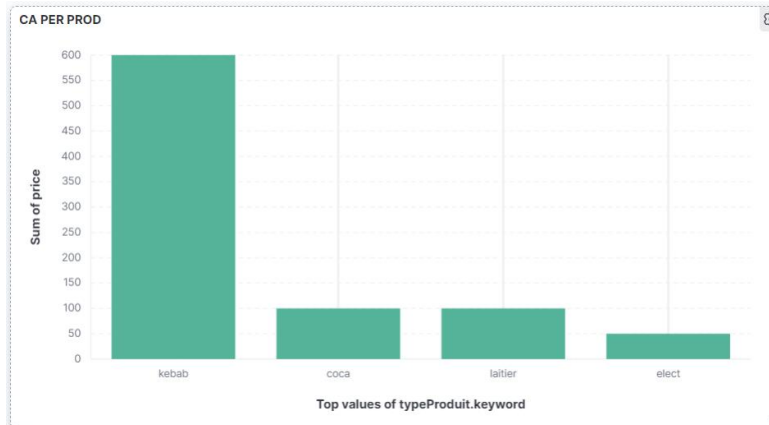
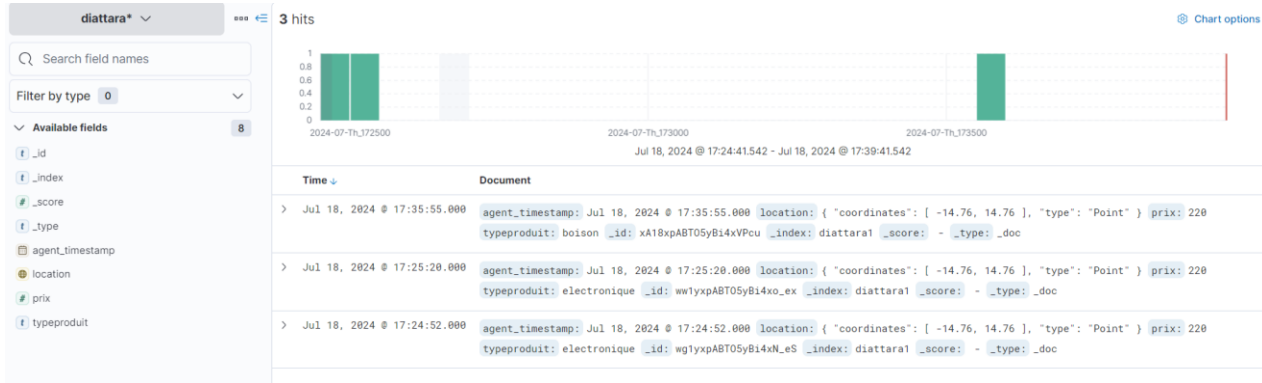
Un dashboard avec un map(chiffre d'affaire par location) et un viz bar ca par type de produit

[https://github.com/idiattara/Spark\\_DIATTARA/blob/main/producer.py](https://github.com/idiattara/Spark_DIATTARA/blob/main/producer.py)

[https://github.com/idiattara/Spark\\_DIATTARA/blob/main/EXO2\\_ELASTIC.xml](https://github.com/idiattara/Spark_DIATTARA/blob/main/EXO2_ELASTIC.xml)

\*

# Exercise2



# Solution Index mapping

- Le **schéma** de données correspond à **un mapping**. Mais concrètement, qu'est-ce que c'est ?
- **Lucene** a besoin, pour effectuer des **recherches**, de savoir comment lire les données.
- Si le schéma n'est pas défini, le mapping sera la structure du premier document inséré
- adresse : [http://mosef02.westeurope.cloudapp.azure.com:9200/my-index-01/\\_search?pretty](http://mosef02.westeurope.cloudapp.azure.com:9200/my-index-01/_search?pretty)

*PS: Le paramètre pretty permet de présenter le résultat de manière présentable.*

# Index mapping

il n'est pas possible de modifier le mapping d'un index une fois qu'il a été instancié (après la première importation). Il faut soit le supprimer, soit en créer un nouveau.

## Les strings:

le type "**string**" est divisé en deux nouveaux types:

- **Text** : qui doit être utilisé pour la recherche full-text
- **Keyword** : qui doit être utilisé pour la recherche par mot-clé

et pour les agrégations (count, ...).

```
"actors" : {  
  "type" : "text",  
  "fields" : {  
    "keyword" : {  
      "type" : "keyword",  
      "ignore_above" : 256  
    }  
  }  
},  
"directors" : {  
  "type" : "text",  
  "fields" : {  
    "keyword" : {  
      "type" : "keyword",  
      "ignore_above" : 256  
    }  
  }  
},  
}
```

# Index template

Un "**template**" est un moyen d'indiquer à Elasticsearch **comment configurer un index lors de sa création**.

Les "templates" sont configurés **avant la création de l'index**, puis lorsqu'un index est créé manuellement ou via l'indexation d'un document, les paramètres du "template" sont utilisés comme base pour la création de l'index.

# Indexation des données dans Elasticsearch

## Bulk API

Permet d'effectuer **plusieurs opérations d'indexation** ou **de suppression** en un seul appel d'API. Cela peut **augmenter** considérablement la **vitesse d'indexation**.

Exemple : (Dataset : [https://github.com/idiattara/data-ELK/blob/main/movies\\_elastic.json](https://github.com/idiattara/data-ELK/blob/main/movies_elastic.json))

```
curl -XPUT -H "Content-Type: application/json" http://mosef02.westeurope.cloudapp.azure.com:9200/_bulk --data-binary @movies_elastic.json
```

←	→	↺	⚠ Non sécurisé   20.101.123.129:8082/_cat/indices?v						
health	status	index	uuid	pri	rep	docs.count	docs.deleted	store.size	pri.store.size
yellow	open	toto	JQJMTT7kR8a9nGX63DUAdQ	1	1	1	0	4.5kb	4.5kb
yellow	open	fournisseur_20211210_064746	-JzmNZIYSnejApcOf2tGzA	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_035543	CV6PBRTuRjQNL5YwdFCNKA	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_041641	FIVI9DfmRrGSgbJF20m-8w	1	1	4	0	5.4kb	5.4kb
green	open	.kibana_task_manager_7.16.0_001	BNnW1UwSTsS50SVxRBeyWg	1	0	17	64411	6.4mb	6.4mb
yellow	open	fournisseur_20211209	a4qnXTzkSjwF23IB9TyquQ	1	1	4	0	5.4kb	5.4kb
green	open	.kibana_7.16.0_001	sNhppQe7TtxqOMVo3nziBA	1	0	25	2	2.3mb	2.3mb
yellow	open	fournisseur_20211210_030332	hgfbFqHtSwa1iHfSb3GgcA	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_070247	mXr77pkxSAMhad4mh7JppA	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_050739	bJ5TOy4gQ4mtqFh7c1MJjg	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_030335	SGWG7F2WSuiZ6oL8o9SCDw	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_032217	19YKf6goTZiBHC294t-Bfg	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_073719	HZpYd4FpRSa6G4EEsNWNwg	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_055608	u8B_NaGcQuuuuFZc1JDkPg	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_063150	AUjK3f2wTImgbD93P_lmsg	1	1	4	0	5.4kb	5.4kb
yellow	open	my-index-01	vbwwgx5MScIwJvoOzuRB-zQ	1	1	1	0	4.8kb	4.8kb
yellow	open	movies	Dng5wyuxSmaqyUeWvXM8qg	1	1	4849	0	4mb	4mb



# Reindex API

Permet de copier des documents d'un index/alias source vers un index/alias destination.

La source et la destination devront être différents

## Exemple :

```
curl -X POST "http://20.101.123.129:8082/_reindex?pretty" -H 'Content-Type: application/json' -d'
```

```
{  
  "source": {  
    "index": "my-index-000001"  
  },  
  "dest": {  
    "index": "my-new-index-000001"  
  }  
}
```

Requête

# Search API

Renvoie les documents JSON qui correspondent à une requête.

curl -X GET [http://20.101.123.129:8082/my-index000001/\\_search?pretty](http://20.101.123.129:8082/my-index000001/_search?pretty)

```
$ curl -X GET http://20.101.123.129:8082/my-index-000001/_search?pretty
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             0         0     15606   0 --:--:-- --:--:-- --:--:-- 15720{
"took" : 0,
"timed_out" : false,
"_shards" : {
  "total" : 1,
  "successful" : 1,
  "skipped" : 0,
  "failed" : 0
},
"hits" : {
  "total" : {
    "value" : 3,
    "relation" : "eq"
  },
  "max_score" : 1.0,
  "hits" : [
    {
      "_index" : "my-index-000001",
      "_type" : "_doc",
      "_id" : "TiOxrn0BdwEUm1C-ImU8",
      "_score" : 1.0,
      "_source" : {
        "@timestamp" : "2021-01-15T13:12:00",
        "message" : "GET /search HTTP/1.1 200 1070000",
        "user" : {
          "id" : "kimchy"
        }
      }
    }
  ]
}
```

# Search API : Exemples

curl -X GET [http://20.101.123.129:8082/movies/\\_search?q=Star+Wars](http://20.101.123.129:8082/movies/_search?q=Star+Wars)

⇒ retourne les films qui contiennent le mot “Star Wars”

curl -X GET [http://20.101.123.129:8082/movies/\\_search?q=fields.actors:Harrison+Ford](http://20.101.123.129:8082/movies/_search?q=fields.actors:Harrison+Ford)

⇒ retourne les films dont le champs “fields.actors” contient le mot “Harrison Ford”

curl -X GET [http://20.101.123.129:8082/movies/\\_search?q=fields.actors:Harrison+Ford&size=20](http://20.101.123.129:8082/movies/_search?q=fields.actors:Harrison+Ford&size=20)

⇒ retourne 20 films dont le champs “fields.actors” contient le mot “Harrison Ford”

curl -X GET [http://20.101.123.129:8082/movies/\\_search?q=fields.title:Star+Wars%20AND%20fields.directors:George+Lucas](http://20.101.123.129:8082/movies/_search?q=fields.title:Star+Wars%20AND%20fields.directors:George+Lucas)

⇒ retourne les films dont le champs “fields.title” contient le mot “Star Wars” et le champs “fields.directors” contient le mot “George Lucas”

# Search API : Examples

```
curl -X GET -H "Content-Type: application/json" 'http://20.101.123.129:8082/movies/_search?pretty' -d '{  
  "query":{  
    "match":{  
      "fields.title":"Star Wars"  
    }  
  }  
}'
```

```
curl -XGET -H "Content-Type: application/json" 'http://20.101.123.129:8082/movies/_search?pretty' -d '  
{"query":{  
  "bool": {  
    "should": [  
      { "match_phrase": { "fields.title": "Star Wars" }},  
      { "match": { "fields.directors": "George Lucas" }}  
    ]  
  }  
}}}'
```

# Exercice

- 1 retrouver les films dont le genre contient "Action" et l'acteur est "Brad Pitt"
- 2 retrouver les films de James Cameron (fields.directors) dont le rang (fields.rank) est inférieur strictement à 1000

# Solution

1-

```
curl -XGET -H "Content-Type: application/json" 'http://20.101.123.129:8082/movies/_search?pretty' -d '
```

```
{"query":{
```

```
  "bool": {
```

```
    "must": [
```

```
      { "match": { "fields.genres": "Action" } },
```

```
      { "match": { "fields.actors": "Brad Pitt" } }
```

```
    ]
```

```
  } } }
```

# Solution

2-

```
curl -XGET -H "Content-Type: application/json" 'http://20.101.123.129:8082/movies/_search?pretty' -d '
```

```
{"query":{
```

```
  "bool": {
```

```
    "must": [
```

```
      { "match_phrase": { "fields.directors": "James Cameron" } },
```

```
      { "range": { "fields.rank": {"lt":1000 } } }
```

```
    ]
```

```
  } } }
```



# Les agrégations

Les agrégations permettent de fournir des statistiques sur le contenu de la base de données.

Exemple d'agrégations : count, avg, sum ...

## Exemple :

```
curl -XGET -H "Content-Type: application/json" 'http://20.101.123.129:8082/movies/_search?pretty' -d '{
  "aggs" : {
    "nb_par_annee" : {
      "terms" : {"field" : "fields.year"}
    }
  }
}'
```

# Exemple

Calculons maintenant la **note moyenne des films**. Pour cela, nous utiliserons la fonction d'agrégation **"avg"** sur la clé **"fields.rating"**

```
curl -XGET -H "Content-Type: application/json" 'http://20.101.123.129:8082/movies/_search?pretty' -d '
```

```
{"aggs" : {
```

```
  "note_moyenne" : {
```

```
    "avg" : {"field" : "fields.rating"}
```

```
  }  
}'
```

# Example

```
curl -X GET "http://20.101.123.129:8082/movies/_search?pretty" -H 'Content-Type: application/json' -d'
```

```
{  
  "aggs": {  
    "avg_rating_per_year": {  
      "terms": {  
        "field": "fields.year"  
      },  
      "aggs": {  
        "avg_rating": { "avg": { "field": "fields.rating" } }  
      }  
    }  
  }  
}
```

# Exercice

Afficher la note moyenne (`fields.rating`) et le rang moyen (`fields.rank`) des films de George Lucas (`fields.directors`)

Attention : il ne faut pas compter les films de George Miller

Hint : `must / match_phrase`

PS : Il y a 6 films de George Lucas

# Solution

```
curl -XGET -H "Content-Type: application/json" 'http://20.101.123.129:8082/movies/_search?pretty' -d '{
```

```
  "query": {
    "bool": {
      "must": {
        "match_phrase": {
          "fields.directors": "George Lucas"
        }
      }
    }
  },
```

```
  "aggs": {
    "note_moyenne": {
      "avg": {
        "field": "fields.rating"
      }
    },
    "rang_moyen": {
      "avg": {
        "field": "fields.rank"
      }
    }
  }
}'
```

# Pagination (Paginate search results)

- Par défaut, **Search API** renvoient les **10 premiers résultats**.
- Pour parcourir un plus grand ensemble de résultats, vous pouvez utiliser les paramètres **from** et **size** de l'API search:
  - Le paramètre **from** définit le nombre de hits à ignorer, par défaut à 0.
  - Le paramètre **size** est le **nombre maximum de hits à renvoyer**.
- Ensemble, ces deux paramètres définissent une **page** de résultats.

## Exemple :

```
curl -X GET -H "Content-Type: application/json" 'http://20.101.123.129:8082/movies/_search?pretty' -d '{  
  "from": 0,  
  "size": 20,  
  "query":{  
    "match":{  
      "fields.title":"Star Wars"  
    }  
  }  
}'
```

# Pagination (Paginate search results)

- il faut **éviter** d'utiliser "from" et "size" pour récupérer des **pages trop profondes** ou pour **demandeur un résultat volumineux**.
- Les requêtes "search" couvrent généralement plusieurs shards. Chaque shard doit charger les hits demandés et les hits de toutes les pages précédentes en mémoire.
- Pour des résultats volumineux, ces opérations peuvent augmenter considérablement **l'utilisation de la mémoire (RAM)** et **du processeur (CPU)**, entraînant une **dégradation des performances** ou des pannes de nœuds.

# Search after

- Vous pouvez utiliser le paramètre **search\_after** pour récupérer la page suivante des résultats en utilisant des valeurs de tri de la page précédente.
- L'utilisation de **search\_after** nécessite que les requêtes aient les **mêmes critères de recherche et de tri**.
- Pour obtenir la première page de résultats, soumettez une requête de recherche avec un **argument de tri**.

## Exemple:

```
curl -X GET "http://20.101.123.129:8082/movies/_search?pretty" -H 'Content-Type: application/json' -d{'
```

```
"size": 100,
```

```
"sort": [
```

```
  {"fields.release_date": "asc"},
```

```
  {"id.keyword": "asc"} ]
```

```
}]'
```



# Search after

Chaque document JSON retournée contient un tableau de valeurs de tri:

```
{
  "_index" : "movies",
  "_type" : "movie",
  "_id" : "3237",
  "_score" : null,
  "_source" : {
    "fields" : {
      "directors" : [
        "William A. Wellman"
      ],
      "release_date" : "1931-04-23T00:00:00Z",
      "rating" : 7.7,
      "genres" : [
        "Crime",
        "Drama"
      ],
      "image_url" : "http://ia.media-imdb.com/images/M/MV5BMTQ3MzI1Njg5M15BMTl5BnBnXkFtZTcwMjM1NDkyMQ@@._V1_SX400_.jpg",
      "plot" : "A young hoodlum rises up through the ranks of the Chicago underworld, even as a gangster's accidental death threatens to spark a bloody mob war.",
      "title" : "The Public Enemy",
      "rank" : 3237,
      "running_time_secs" : 4980,
      "actors" : [
        "James Cagney",
        "Jean Harlow",
        "Edward Woods"
      ],
      "year" : 1931
    },
    "id" : "tt0022286",
    "type" : "add"
  },
  "sort" : [
    -1221091200000,
    "tt0022286"
  ]
}
```

# Search after

- Pour obtenir la page de résultats suivante, relancez la recherche précédente en utilisant **les valeurs de tri du dernier résultat** comme argument search\_after.
- Les **arguments de requête et de tri de la recherche doivent rester inchangés**.

## Exemple:

```
curl -X GET "http://20.101.123.129:8082/movies/_search?pretty" -H 'Content-Type: application/json' -d'{ "size": 100,
  "sort": [
    {"fields.release_date": "asc"},
    {"id.keyword": "asc"}
  ],
  "search_after": [
    -406512000000,
    "tt0050419"
  ]
}'
```

# Exemple avec le package python “elasticsearch”

“elasticsearch” est un package python pour interagir avec Elasticsearch

Lien : <https://elasticsearch-py.readthedocs.io/en/v7.10.1/>

```
pip install elasticsearch
```

Ou

```
python -m pip install elasticsearch
```

# Example

```
from elasticsearch import Elasticsearch
import datetime

es = Elasticsearch([{'host': '20.188.37.241', 'port': 9200}])
doc = {'author': 'kimchy', 'text': 'Elasticsearch: cool. bonsai cool.', 'timestamp':
datetime.datetime.now().strftime( '%d-%m-%Y')}
res = es.index(index="test-index", id=1, body=doc)
print(res)
print(res['result'])

res = es.get(index="test-index", id=1)
print(res['_source'])
#Performs the refresh operation in one or more indices.
es.indices.refresh(index="test-index")

res = es.search(index="test-index", body={"query": {"match_all": {}}})
print("Got %d Hits:" % res['hits']['total']['value'])
for hit in res['hits']['hits']:
    print("(%(timestamp)s %(author)s: %(text)s" % hit["_source"])
```

# Exercice : requêter les données

- Ecrire un **script python** qui permet de **requêter toutes les données** de l'index **“movies”** et de les **écrire dans un fichier texte**.
- Il faut utiliser le mécanisme **“search\_after”** avec le **package python “elasticsearch”**.
- La **taille d'une page** ne doit pas dépasser **1000 documents**.

# Solution

```
from elasticsearch import Elasticsearch

es = Elasticsearch([
    {'host': '20.188.37.241', 'port': 9200}
])
es_index = "movies"
res = es.count(index=es_index)
size = res['count']
print(size)
body = {"size": 1000,
        "sort": [
            {"fields.release_date": "asc"},
            {"id.keyword": "desc"}
        ]
    }

result = es.search(index=es_index, body=body)
print(result)
bookmark = [result['hits']['hits'][-1]['sort'][0], str(result['hits']['hits'][-1]['sort'][1])]
print(bookmark)
```

# Solution

```
body1 = {"size": 1000,
         "search_after": bookmark,
         "sort": [
             {"fields.release_date": "asc"},
             {"id.keyword": "desc"}
         ]}

while len(result['hits']['hits']) < size:
    res = es.search(index=es_index, body=body1)
    for el in res['hits']['hits']:
        result['hits']['hits'].append(el)
    bookmark = [res['hits']['hits'][-1]['sort'][0], str(result['hits']['hits'][-1]['sort'][1])]
    body1 = {"size": 1000,
            "search_after": bookmark,
            "sort": [
                {"fields.release_date": "asc"},
                {"id.keyword": "desc"}
            ]}

the_file = open('my_movies.txt', 'a')

for element in result['hits']['hits']:
    the_file.write(str(element['_source']) + '\n')
```

# Best pratics: Choix number Shard

Evitez des partitions trop grandes, car ces dernières peuvent affecter négativement la capacité du cluster à se remettre d'une défaillance. Bien qu'il n'y ait pas de limite concrète à la taille d'une partition(shard), on mentionne souvent une limite de **50 Go**, car c'est ce qui semble fonctionner pour de nombreux cas d'utilisation

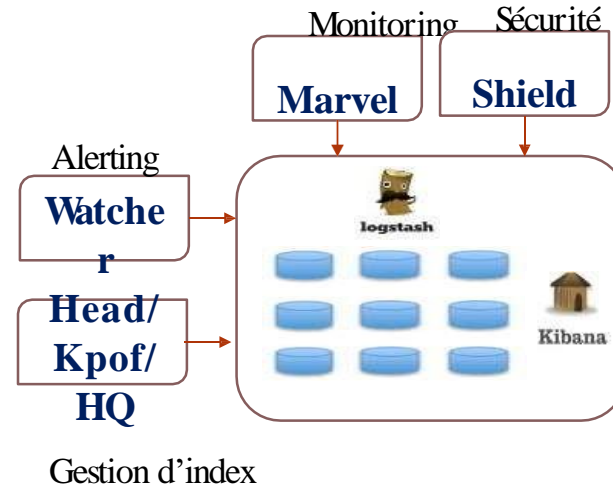


## Plugin

- Shield :  
Protection par login / mot de passe  
Restrictions au niveau des documents et des champs
- Watcher :  
Une extension permettant de gérer des alertes et des notifications :
- Marvel:  
permet de monitorer, diagnostiquer un cluster ES

**X-Pack:** Regroupe les 3 fonctionnalité  
X-Pack ajoutent des fonctions d'alerte, de sécurité, de monitoring, de reporting, de graphe ou de ML au moteur de recherche Elasticsearch.

Certaines sont gratuites, d'autres payantes.  
Elastic a décidé d'en ouvrir le code pour accroître l'engagement de la communauté autour de ces composantes.



## Best pratics:Les index et les partitions sont-ils indépendants ?

Pour chaque index Elasticsearch, les informations relatives au mapping et à l'état sont stockées dans l'état du cluster. Elles sont gardées en mémoire pour un accès rapide. L'utilisation d'un grand nombre d'index et de partitions dans un cluster peut donc entraîner un état de cluster volumineux, surtout si les mappings sont conséquents. Tout ceci peut devenir long à mettre à jour, étant donné que toutes les modifications doivent être effectuées dans un seul thread afin de garantir la cohérence avant que les changements ne soient distribués à travers le cluster.

CONSEIL : les petites partitions produisent des petits segments, ce qui augmente la surcharge. Efforcez-vous de maintenir la taille moyenne d'une partition entre au moins quelques Go et quelques dizaines de Go. Pour les cas d'utilisation avec des données temporelles, les partitions font généralement entre 20 Go et 40 Go.

CONSEIL : étant donné que la surcharge par partition dépend du nombre de segments et de leur taille, forcer la fusion de plus petits segments en de plus gros segments peut réduire la surcharge et améliorer les performances de recherche. Dans l'idéal, cette opération doit être effectuée une fois que les données ne sont plus écrites dans l'index. Sachez que cette opération est gourmande et devrait idéalement être effectuée hors des heures de pics d'indexation.


CONSEIL : le nombre de partitions pouvant être contenues dans un nœud sera proportionnel à la quantité de mémoire disponible, mais Elasticsearch n'applique pas de limite fixe. Une bonne règle générale est de s'assurer que le nombre de partitions par nœud reste en dessous de 20 par Go de mémoire configurée. Un nœud avec une mémoire de 30 Go devrait donc contenir un maximum de 600 partitions, mais plus vous êtes en dessous de la limite, mieux c'est. Cette opération permet généralement de préserver l'intégrité du cluster.

# Kibana

- Kibana est un **client pour Elasticsearch** qui fournit à l'utilisateur une **interface graphique** (UI) accessible par un navigateur web.
- Cette UI permet à l'utilisateur de réaliser des **recherches**, de **visualiser** des documents individuels et d'agréger des résultats dans des **graphes** et des **diagrammes**.
- Fait partie de la suite Elastic
- URL Kibana : <http://20.101.123.129:5601/>

Non sécurisé | 20.101.123.129:5601/app/home#/

D


 elastic

Search Elastic

D


Home

# Welcome home




## Enterprise Search

Create search experiences with a refined set of APIs and tools.




## Observability

Consolidate your logs, metrics, application traces, and system availability with purpose-built UIs.



## Security

Prevent, collect, detect, and respond to threats for unified protection across your infrastructure.



## Analytics

Explore, visualize, and analyze your data using a powerful suite of analytical tools and applications.

# Kibana et Elasticsearch

Les objets créés par Kibana sont tous stockés dans un index Elasticsearch

Non sécurisé | 20.101.123.129:8082/\_cat/indices

green	open	.kibana_7.16.0_001	sNhppQe7TxqOMVo3nziiBA	1	0	25	2	2.3mb	2.3mb
yellow	open	fournisseur_20211210_070247	mXr77pkxSAMhad4mh7JppA	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_030332	hgFBfQhTSwa1iHfSb3GgcA	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_050739	bJ5TOy4gQ4mtqFh7c1MJjg	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_030335	SGWG7F2WSuiZ6oL8o9SCDw	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_032217	l9YKf6goTZiBHC294t-Bfg	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_073719	HZpYd4FpRSa6G4EEsNWNwg	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_055608	u8B_NaGcQuuuuFZc1JDkPg	1	1	4	0	5.4kb	5.4kb
yellow	open	fournisseur_20211210_063150	AUjK3f2wTImgbD93P_lmsg	1	1	4	0	5.4kb	5.4kb
yellow	open	my-index-01	vbwvgx5MScWjvoOzuRB-zQ	1	1	1	0	4.8kb	4.8kb
yellow	open	movies	Dng5wyuxSmaqyUeWvXM8qg	1	1	4849	0	4mb	4mb

# Index-pattern

Un pattern qui permet de matcher un ou plusieurs index Elasticsearch.

## **Exemple :**

- Index pattern “movies” permet de matcher l’index Elasticsearch “movies”
- Index pattern “my-\*” permet de matcher les index Elasticsearch “my-index-000001” et “my-new-index-000001”



Stack Management

Index patterns



## Management

### Ingest ⓘ

Ingest Pipelines

### Data ⓘ

Index Management

Index Lifecycle Policies

Snapshot and Restore

Rollup Jobs

Transforms

Remote Clusters

### Alerts and Insights ⓘ

Rules and Connectors

Reporting

Machine Learning Jobs

# Index patterns

[+ Create index pattern](#)

Create and manage the index patterns that help you retrieve your data from Elasticsearch.

Pattern ↑




[movies\\*](#)



Rows per page: 10 ▾




< 1 >


# Discover

← → ↻ ⚠ Non sécurisé | 20.101.123.129:5601/app/discover#/?\_g=(filters:!(),refreshInterval:(pause:!t,value:0),time:(from:now-15m,to:now))&\_a=(columns:!(),filter:...

 Search Elastic  

≡  Discover ▾ Options New Open Share Inspect  Save

 ▾ Search  KQL  Refresh









 + Add filter

**movies\*** ▾ 4,849 hits

🔍 Search field names

Filter by type 0 ▾

▼ Available fields 17

-  \_id
-  \_index
-  # \_score
-  \_type
-  fields.actors
-  fields.directors
-  fields.genres
-  fields.image\_url

**Document**

> **fields.actors:** Joseph Gordon-Levitt, Scarlett Johansson, Julianne Moore **fields.directors:** Joseph Gordon-Levitt  
**fields.genres:** Comedy, Drama **fields.image\_url:** http://ia.media-imdb.com/images/M/MV5BMTQxNTc3NDM2MF5BMl5BanBnXkFtZTcwNzQ5NTQ3ODQ@.\_V1\_SX400\_.jpg **fields.plot:** A New Jersey guy dedicated to his family, friends, and church, develops unrealistic expectations from watching porn and works to find happiness and intimacy with his potential true love. **fields.rank:** 1 **fields.rating:** 7.4 **fields.release\_date:** Jan 18,

> **fields.actors:** Daniel Brühl, Chris Hemsworth, Olivia Wilde **fields.directors:** Ron Howard **fields.genres:** Action, Biography, Drama, Sport **fields.image\_url:** http://ia.media-imdb.com/images/M/MV5BMTQyMDE0MTY0OV5BMl5BanBnXkFtZTcwMjI2OTI0OQ@.\_V1\_SX400\_.jpg **fields.plot:** A re-creation of the merciless 1970s rivalry between Formula One rivals James Hunt and Niki Lauda. **fields.rank:** 2 **fields.rating:** 8.3  
**fields.release\_date:** Sep 2, 2013 @ 02:00:00.000 **fields.running\_time\_secs:** 7,380 **fields.title:** Rush **fields.year:** 2,013

> **fields.actors:** Hugh Jackman, Jake Gyllenhaal, Viola Davis **fields.directors:** Denis Villeneuve **fields.genres:** Crime,



# Exercice

1- Re-indexer l'index elasticsearch "movies" dans un autre index sous le nom "prénom\_movies" (exemple : ibrahima\_movies)

2- Créer un index pattern kibana qui permet de matcher l'index Elasticsearch "prénom\_movies" (exemple : ibrahima\_movies)

3- Créer un dashboard qui contient un pie chart et Bar horizontale qui permettent d'afficher le nombre de films par "directors"

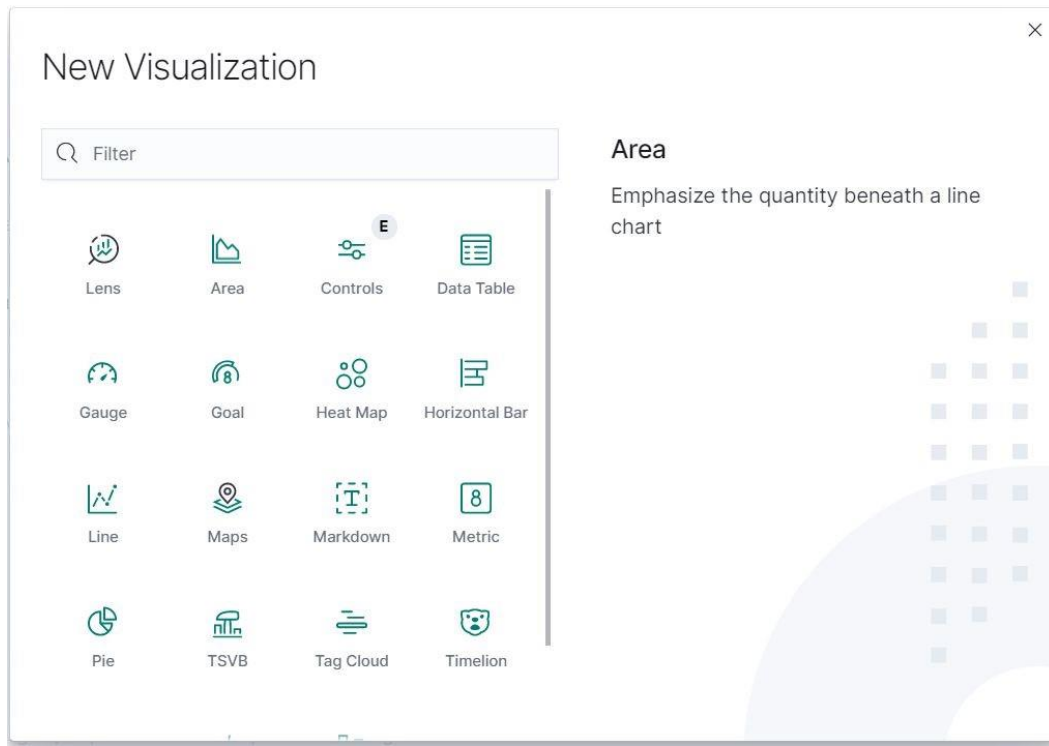
# Solution

## ReindexAPI:

```
curl -X POST "http://20.188.37.241:9200/_reindex?pretty" -H 'Content-Type: application/json' -d'
{
  "source": {
    "index": "movies"
  },
  "dest": {
    "index": "ibrahima_movies"
  }
}
'
```

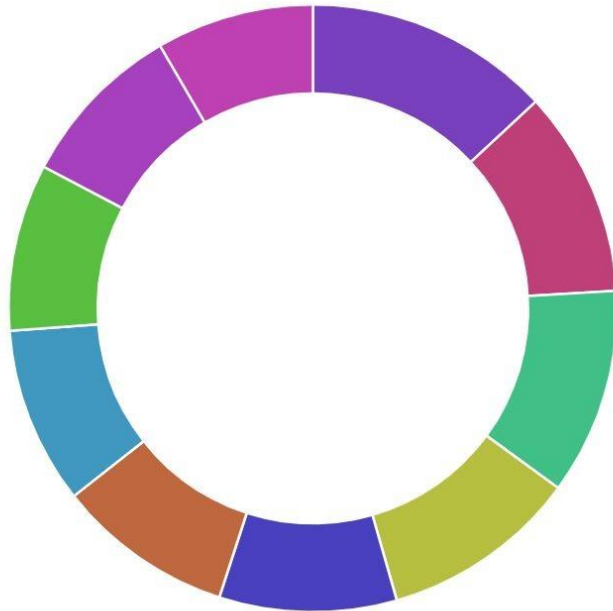
# Visualization

Des objets qui permettent d'appliquer des agrégations sur les données.



# Exemple

Créer un pie chart qui permet d'afficher le nombre de films par “directors”



- Steven Spielberg
- Clint Eastwood
- Ron Howard
- Alfred Hitchcock
- Martin Scorsese
- Ridley Scott
- Woody Allen
- Robert Rodriguez
- Steven Soderbergh
- Ethan Coen

**movies**

Data Options

**Buckets**

Split slices

Aggregation

Terms

Field

fields.directors.keyword

Order by

Metric: Count

Order

Descending

Size

10

☐ Group other values in separate bucket

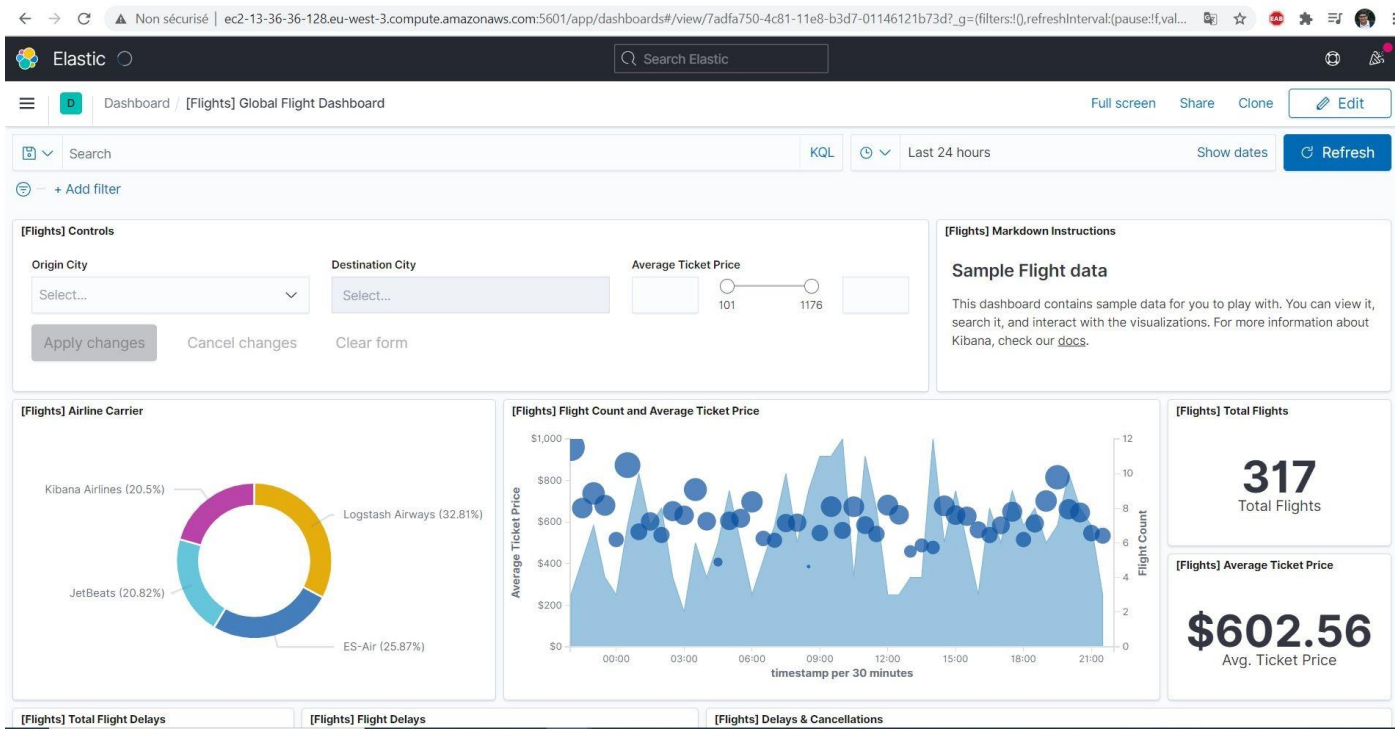
☐ Show missing values

Custom label


Discard Update

# Dashboard

Un dashboard Kibana contient plusieurs types d'objets : discover, visualizations ...



# Dev tools

 Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help

1 GET movies/\_mapping  
2

200 - OK 74 ms

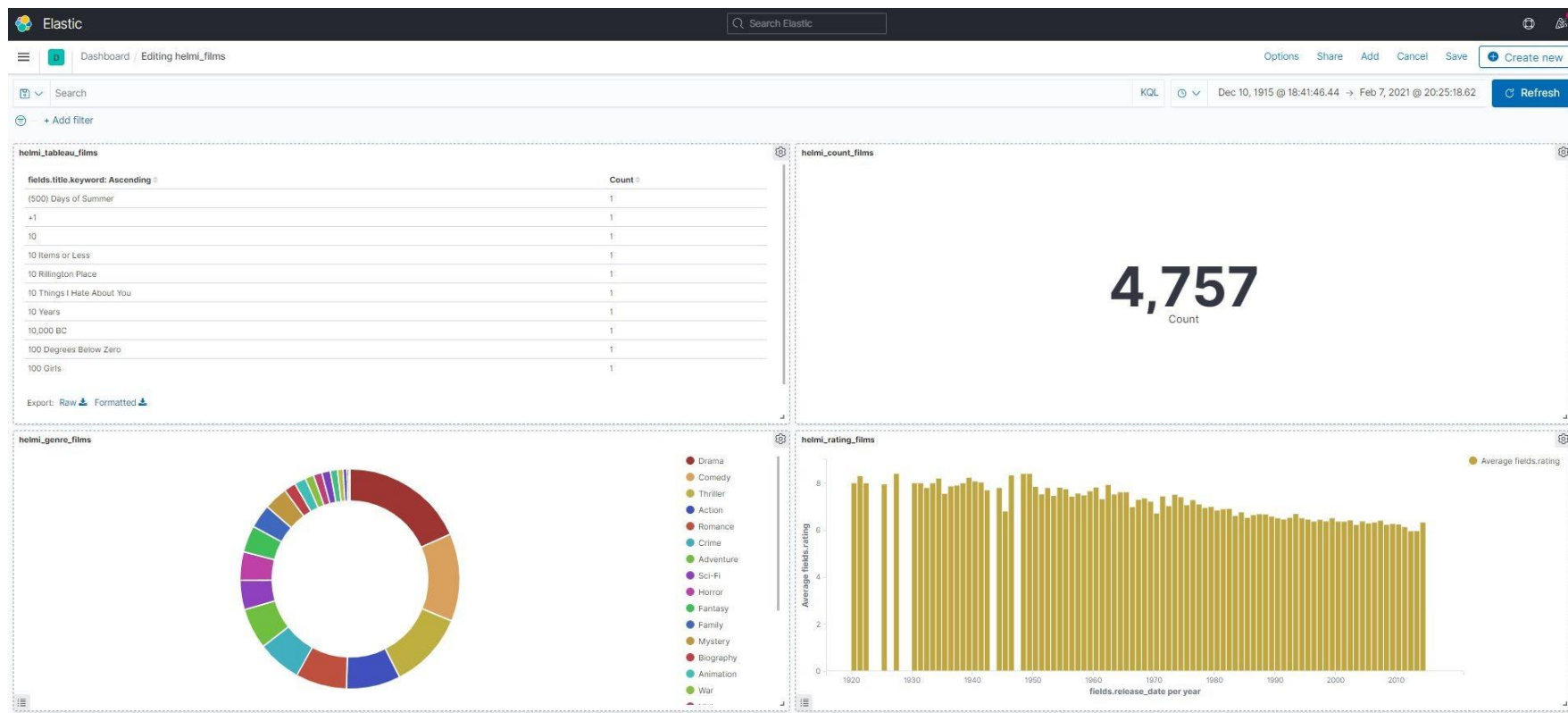
```
1 {  
2   "movies" : {  
3     "mappings" : {  
4       "properties" : {  
5         "fields" : {  
6           "properties" : {  
7             "actors" : {  
8               "type" : "text",  
9               "fields" : {  
10                "keyword" : {  
11                  "type" : "keyword",  
12                  "ignore_above" : 256  
13                }  
14              }  
15            },  
16            "directors" : {  
17              "type" : "text",  
18              "fields" : {  
19                "keyword" : {  
20                  "type" : "keyword",  
21                  "ignore_above" : 256  
22                }  
23              }  
24            },  
25            "genres" : {  
26              "type" : "text",  
27              "fields" : {  
28                "keyword" : {  
29                  "type" : "keyword",  
30                  "ignore_above" : 256  
31                }  
32              }  
33            },  
34            "image_url" : {  
35              "type" : "text"
```

# Exercice

Créer un dashboard qui contient :

- Une visualisation de type “**tableau**” contenant les **titres des films**
- Une **visualisation** de type “**metric**” qui affiche le **nombre des films** avec le **genre “action”**
- Une **visualisation** de type “**pie chart**” qui affiche les **différents genre des films**
- Une **visualisation** de type “**vertical bar chart**” qui affiche l’évolution dans le temps de la moyenne du “average ratings” par an

# Solution





# ELK

## Connection ELK

### ❖ **ElasticSearch [java]:**

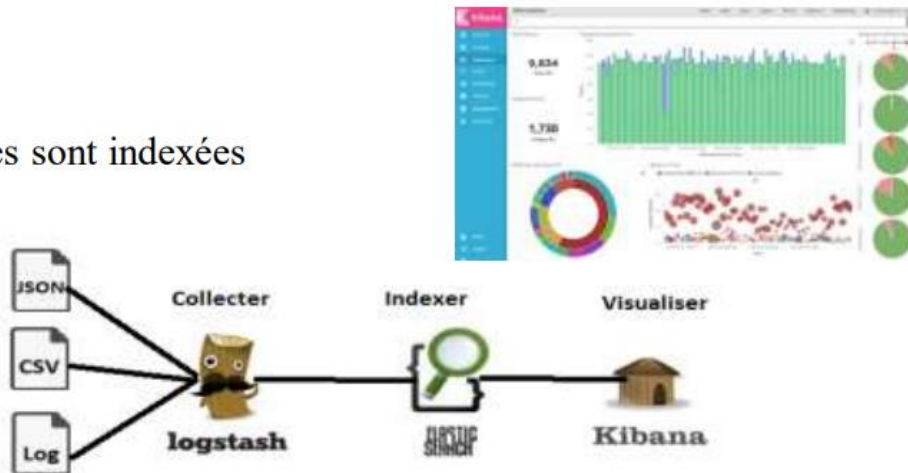
- un outil de stockage où toutes les données sont indexées
- Fonctionne en cluster

### ❖ **Logstash [java & ruby]:**

- Collecter, parser, filtrer des logs
- Transforme tout en JSON
- Stocker dans elasticsearch

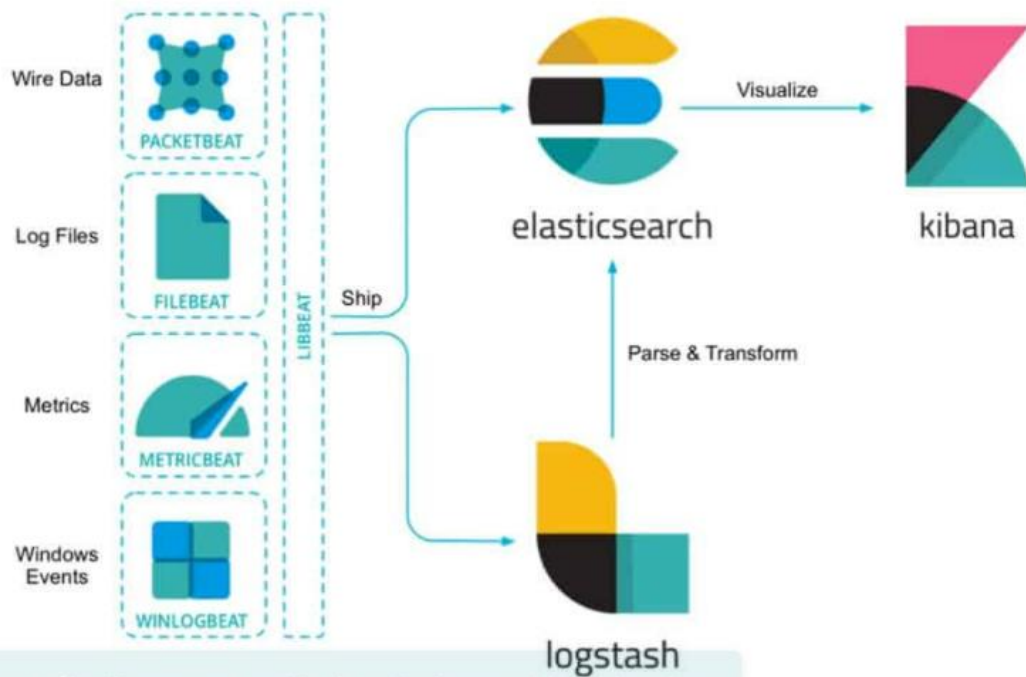
### ❖ **Kibana [Javascript]:**

- Une interface web permettant de rechercher des infos stockées par Logstash dans ElasticSearch
- Créer des dashboard(histogrammes, barres, cartes)



# ELK

## Elastic (ELK) Stack Architecture



ici pour sélectionner une partie de cette image et la rechercher