Rayan HADDAD **Master 2 DATASCALE**
Raphaël RAIMBAUD REMAZEILLES

Projet d'Indexation et Visualisation de Données: Compte-Rendu

INTRODUCTION

Le but de ce projet était de créer une pipeline permettant de collecter des données depuis une API publique, de les indexer dans ElasticSearch et de les consulter. Nous allons ici expliquer les étapes en détails ainsi qu'indiquer les fichiers importants pour ce projet.

Les données utilisées correspondent aux récentes éditions utilisateurs des pages de Wikipédia. La documentation de cet API est accessible à la page suivante :

https://www.mediawiki.org/wiki/API:RecentChanges

Le fichier **requirements.txt** contient les bibliothèques utilisées pour notre environnement Python. Nous avons utilisé la version 2.12-3.2.0 de Kafka et la version 3.5.4 de Spark.

KAFKA

Avant toute chose, nous avons créer un topic Kafka nommé 'wikipedia_changes' avec la commande suivante après s'être placé dans le dossier de kafka :
bin/kafka-topics.sh --create --topic wikipedia_changes --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

Aprés ça, nous avons crée un script Python qui collecte les données de manière régulière et les envoie dans le topic Kafka : **kafka-producer.py**. A partir des paramètres 'interval' et 'limit' de la fonction 'collect_data', il est possible de changer respectivement le nombre de secondes entre chaque requête et le nombre d'objets renvoyés par la requête.

Avant d'exécuter le script, nous avons aussi utilisé Logstash afin de consommer les données Kafka et les indexer dans ElasticSearch.

LOGSTASH

Le fichier **wikipedia_changes.conf** contient la configuration Logstash utilisée. Nous avons décidé de ne garder que quelques champs clés qui nous semblaient intéressants et nous avons ensuite indiqué qu'il faut indexer les données dans l'index 'wikipedia_changes'.

Les mappings pour l'index se trouvent dans le fichier **wikipedia_changes-mappings.json**. Nous avons décidé de traiter les titres d'article comme du texte avec la possibilité d'interroger comme un mot-clé. Nous avons aussi ajouter un analyseur personnalisé afin de pouvoir faire des requêtes préfixe sur le titre des articles.

Nous avons crée l'index avec la commande suivante :

curl -H ‘Content-Type : application/json’ -XPUT ‘127.0.0.1:9200/wikipedia_changes’ --data-binary @wikipedia_changes-mappings.json

Une fois l’index crée, on peut démarrer l’indexation en démarrant Zookeeper, Kafka, Logstash et en exécutant le script **kafka-producer.py**. L’exécution peut-être stoppée avec Ctrl+C.

REQUETES ELASTICSEARCH

Voici d’abord un exemple des données indexés :


```
{
  "_index" : "wikipedia_changes",
  "_id" : "tB8yKpUBEPgRgFyj9VNr",
  "_score" : 1.0,
  "_source" : {
    "comment" : "",
    "ns" : 1,
    "timestamp" : "2025-02-21T20:28:56Z",
    "type" : "edit",
    "title" : "Talk:The Odd of Love",
    "user" : "Nkon21"
  }
},
{
  "_index" : "wikipedia_changes",
  "_id" : "sh8yKpUBEPgRgFyj9VNr",
  "_score" : 1.0,
  "_source" : {
    "comment" : "/* Reception */",
    "ns" : 0,
    "timestamp" : "2025-02-21T20:28:54Z",
    "type" : "edit",
    "title" : "Alice in Chains (album)",
    "user" : "MFTP Dan"
  }
},
{
  "_index" : "wikipedia_changes",
  "_id" : "sx8yKpUBEPgRgFyj9VNr",
  "_score" : 1.0,
  "_source" : {
    "comment" : "",
    "ns" : 0,
    "timestamp" : "2025-02-21T20:28:54Z",
    "type" : "edit",
    "title" : "Mr. Miyagi",
    "user" : "2600:4040:7122:1A00:ADA9:99BF:6C5F:8F0"
  }
},
```

Afin d'observer les données, nous avons effectué 5 requêtes. Le résulat de chaque requête est contenu dans le dossier **resultat_requetes**.

Requête 1 : Les documents dont le commentaire pour le changement contient le mot 'added'

```
curl -H "Content-Type: application/json" -XGET "localhost:9200/wikipedia_changes/_search?pretty" -d '
{
	"query": {
		"match": {
			"comment": "added"
		}
	}
}
'
```

Requête 2 : Le nombre de documents par type de changement

```
curl -H "Content-Type: application/json" -XGET "localhost:9200/wikipedia_changes/_search?pretty" -d '
{
	"aggs": {
		"types_of_changes": {
			"terms": {
				"field": "type"
			}
		}
	}
}
'
```

Requête 3 : Les documents indiquant un changement pour un article dont le titre commence par 'talk'

```
curl -H "Content-Type: application/json" -XGET "localhost:9200/wikipedia_changes/_search?pretty" -d '
{
	"query": {
		"prefix": {
			"title.prefix": {
				"value": "talk"
			}
		}
	}
}
'
```

Requête 4 : Les documents indiquant un changement par un utilisateur dont le nom est 'olcanoguy' en acceptant un caractère erroné ou manquant.

```
curl -H "Content-Type: application/json" -XGET "localhost:9200/wikipedia_changes/_search?pretty" -d '
{
```

```
        "query": {
                "fuzzy": {
                        "user": {
                                "value": "olcanoguy",
                                "fuzziness": 1
                        }
                }
        }
}
'
```

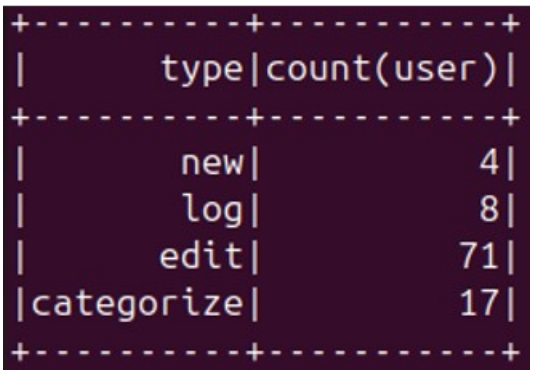
Requête 5 : Les documents correspondant au changement s d'hier et d'aujourd'hui et dont le type de changement est « edit ».

```
curl -H "Content-Type: application/json" -XGET "localhost:9200/wikipedia_changes/_search?pretty" -d '
{
        "query": {
                "bool": {
                        "filter": [ {
                                "range": {
                                        "timestamp": {
                                                "gte": "now-1d/d",
                                                "lte": "now/d"
                                        }
                                }
                        },
                        {
                                "term": {
                                        "type": {
                                                "value": "edit"
                                        }
                                }
                        } ]
                }
        }
}
'
```

SPARK

Nous avons décidé d'effectuer un calcul avec Spark, spécifiquement nous avons crée un script **spark-script.py** qui effectue ce calcul à l'aide de la bibliothèque 'pyspark'. Pour exécuter ce script, il faut utiliser la commande **spark-submit** depuis le dossier contenant spark :
bin/spark-submit --packages org.elasticsearch:elasticsearch-spark-30_2.12:8.7.0 spark-script.py

Ici, nous avons décidé de calculer le nombre d'utilisateurs par type de changement, le résultat que nous obtenont est le suivant :



```
+----------+-----------+
|      type|count(user)|
+----------+-----------+
|       new|          4|
|       log|          8|
|      edit|         71|
|categorize|         17|
+----------+-----------+
```

Nous avons donc une pipeline fonctionnel nous permettant de collecter continuellement des données en ligne puis de faire des agrégats sur ces données.