Chapitre1 Nifi

Mr DIATTARA Ibrahima
Consultant Senior Big Data
Professeur Cloud Data Engineer à l'Université Paris Sorbonne

Sommaire

Définitions

Big Data : Principe des « 4V » (Volume, Variety, Velocity et Valeur)

Le Big Data est une nouvelle génération de technologies et d'architectures pour extraire de la valeur à partir de très large volumes et variétés (+sources , +types(structurées , semi structuré et non structurées) de données avec une grande vélocité de collecte , de transformation et d'analyse

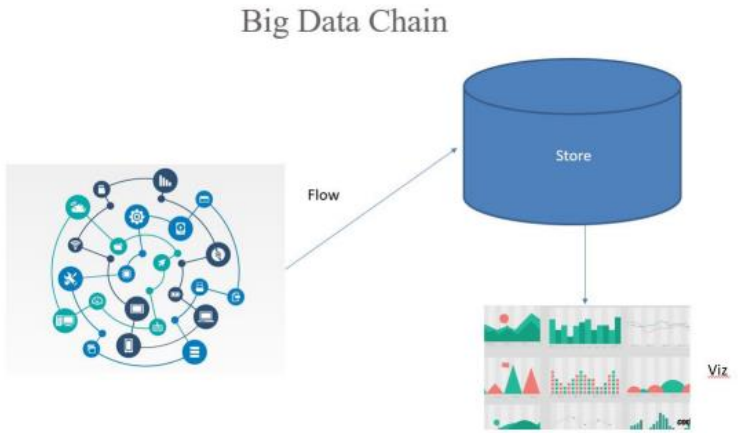
Data-Lake est une zone logique de données ou se déverse l'ensemble des données de l'entreprise pour répondre aux besoins dits Big-Data (agrégation, corrélation, traitement visualisation ..).

Data warehouses: Les données ne sont pas chargées dans l'entrepôt de données tant que leur utilisation n'a pas été définie.

Cluster: ensemble de machine avec master et une ou plusieurs esclaves

Datacenter :lieu où se trouvent différents équipements électroniques, des ordinateurs, des systèmes **de** stockage (rack, …)

Rack/Baie une armoire technique qui centralise des éléments (Disk, …)



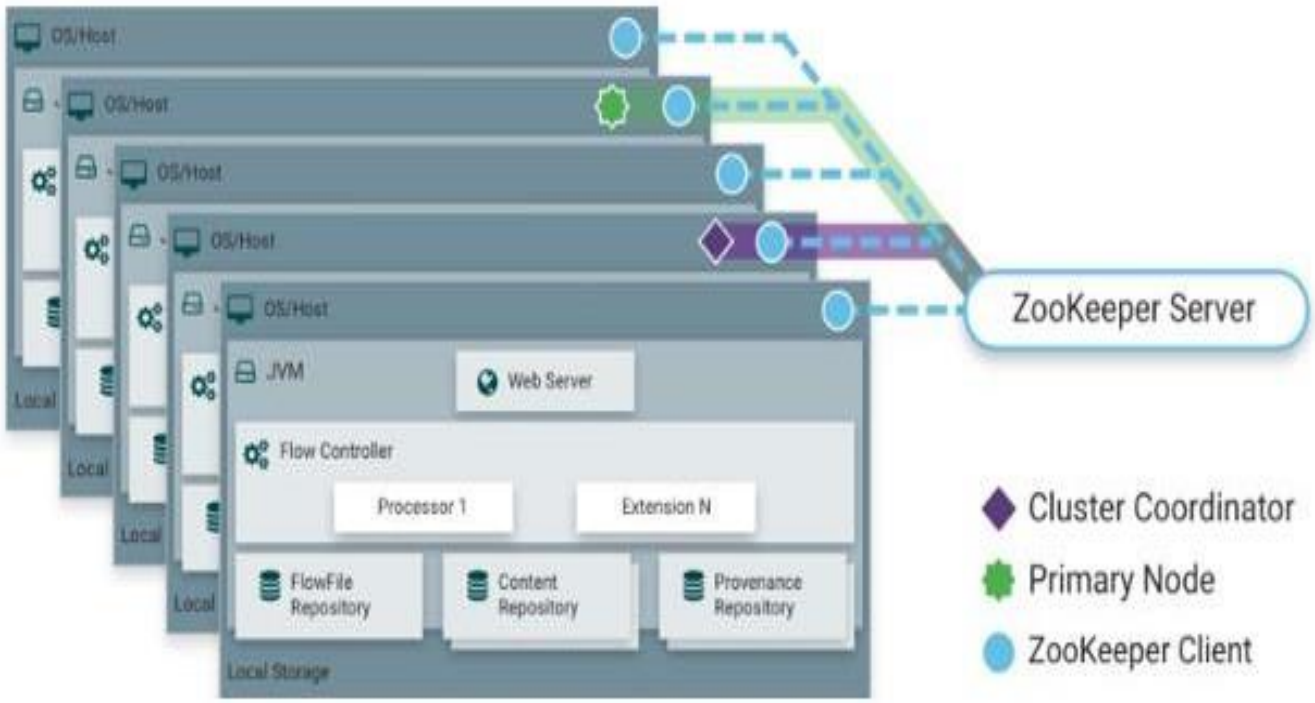
Introduction

- NiFi est un logiciel libre de gestion de **flux de données** développé en java ,il permet de gérer et d'automatiser des flux de données entre plusieurs systèmes informatiques, à partir d'une interface web et dans un environnement distribué (plusieurs nodes)
- Basé sur le paradigme de programmation **flow-based programming**, NiFi fournit une interface web qui permet de construire un flux de données en Drag et Drog. Ainsi, il est possible de définir, de contrôler en temps réel, et d'une certaine manière, de sécuriser l'acheminement de données.
- Apache NiFi assure l'intégralité du flux de données, il est tolérant aux pannes, est scalable et a été conçu pour gérer de gros volumes de données en temps réel.
- Apache NiFi est compatible avec Kerberos qui assure l'authentification et avec Apache Ranger qui permet la sécurité des autorisations d'accès
- Un mode https qui permet d'encrypter les données en transit

Comment utiliser Nifi

NiFi peut tourner dans une JVM en mode local sur le système d'exploitation hôte (Windows, Linux ou Mac) mais on peut aussi déployer Apache NiFi en mode cluster, un de ses grands atouts une fois en production.

Sachez qu'en mode cluster, vous aurez besoin d'Apache Zookeeper pour la gestion de la configuration afin d'assurer la haute disponibilité des services



Coordinateur Node

- Il joue le rôle du **master** sans en l'être
- **Gestion de la Configuration** : Le nœud coordinateur est responsable de la gestion de la configuration globale du cluster NiFi. Cela inclut la gestion des composants, des paramètres de sécurité, et des configurations spécifiques aux processeurs et aux connexions. Autrement dit Il fournit aussi la configuration nécessaire aux nodes autrement dit il est responsable de réplication des modifications sur les autre nodes
- **Coordination des Tâches** : Le coordinateur orchestre la distribution des tâches de traitement des flux de données aux différents nœuds du cluster. Il assure également la cohérence et la synchronisation des données entre les nœuds pour garantir un traitement efficace et sans perte.
- **Haute Disponibilité et Tolérance aux Pannes** : En cas de défaillance d'un nœud du cluster, le coordinateur est chargé de la gestion de la haute disponibilité. Il peut rééquilibrer les charges et reconfigurer dynamiquement les connexions pour maintenir la continuité du traitement des flux de données.

ZooKeeper

Rôle : ZooKeeper assure la coordination entre les nœuds du cluster, garantissant que chaque nœud a une vue cohérente de l'état du cluster. Cela inclut la gestion des élections de nœuds leaders (Coordinator Node) et la synchronisation des configurations de flux de données

Le Cluster Coordinator utilise ZooKeeper pour partager et synchroniser les informations sur l'état du cluster et les répartitions des tâches entre les nœuds

Stockage des Informations : ZooKeeper stocke des informations critiques, y compris la configuration du cluster, l'état des nœuds (actif, inactif), et les métadonnées sur les tâches en cours d'exécution.

Lorsqu'une tâche (ou flow file) est assignée à un nœud pour exécution, cette information est enregistrée dans ZooKeeper. Cela inclut les détails de la tâche et l'identifiant du nœud assigné

Election coordinateur

Apache **ZooKeeper** élit un nœud unique en tant que coordinateur du cluster,

1. **Enregistrement initial :** Lorsqu'un nœud NiFi démarre, il enregistre son existence auprès de ZooKeeper.
2. **Signaux de santé (heartbeat)** : Une fois enregistré, chaque nœud envoie périodiquement des signaux de santé à ZooKeeper pour indiquer qu'il est toujours opérationnel..
3. **Znodes Éphémères Séquentiels** : Lorsqu'un nœud NiFi rejoint le cluster, il crée un znode éphémère séquentiel dans un chemin spécifique dédié à l'élection du coordinateur. Chaque znode éphémère séquentiel se voit attribuer un ID unique séquentiel par ZooKeeper.
4. **Élection basée sur l'ID le plus bas** : ZooKeeper garantit que les znodes éphémères séquentiels sont numérotés dans l'ordre de leur création. NiFi utilise cette séquence pour déterminer quel nœud a l'ID de znode le plus bas. Celui-ci est alors élu comme coordinateur.

Détecter les pannes : Si ZooKeeper cesse de recevoir des heartbeat d'un nœud particulier pendant une période définie (typiquement quelques secondes à quelques minutes, selon la configuration), il peut considérer ce nœud comme défaillant. À ce stade, ZooKeeper peut déclencher des actions telles que la réélection d'un nouveau coordinateur (si le nœud défaillant était le coordinateur principal) ou simplement mettre à jour les informations de cluster pour refléter l'état actuel.

Primary Node :

Rôle:
Le primary node est utilisé pour exécuter des processors isolés

Processus d'élection

- Apache NiFi utilise un algorithme de consensus pour élire un nœud primaire au sein de son cluster. L'algorithme de consensus utilisé par NiFi est basé sur un vote majoritaire (Majority Vote)

- Lors du démarrage ou en cas d'indisponibilité du nœud primaire , les nœuds s'envoient mutuellement des messages pour indiquer leur disponibilité et participer à l'élection.

- chaque nœud évalue la qualité de la communication avec le nœud candidat.

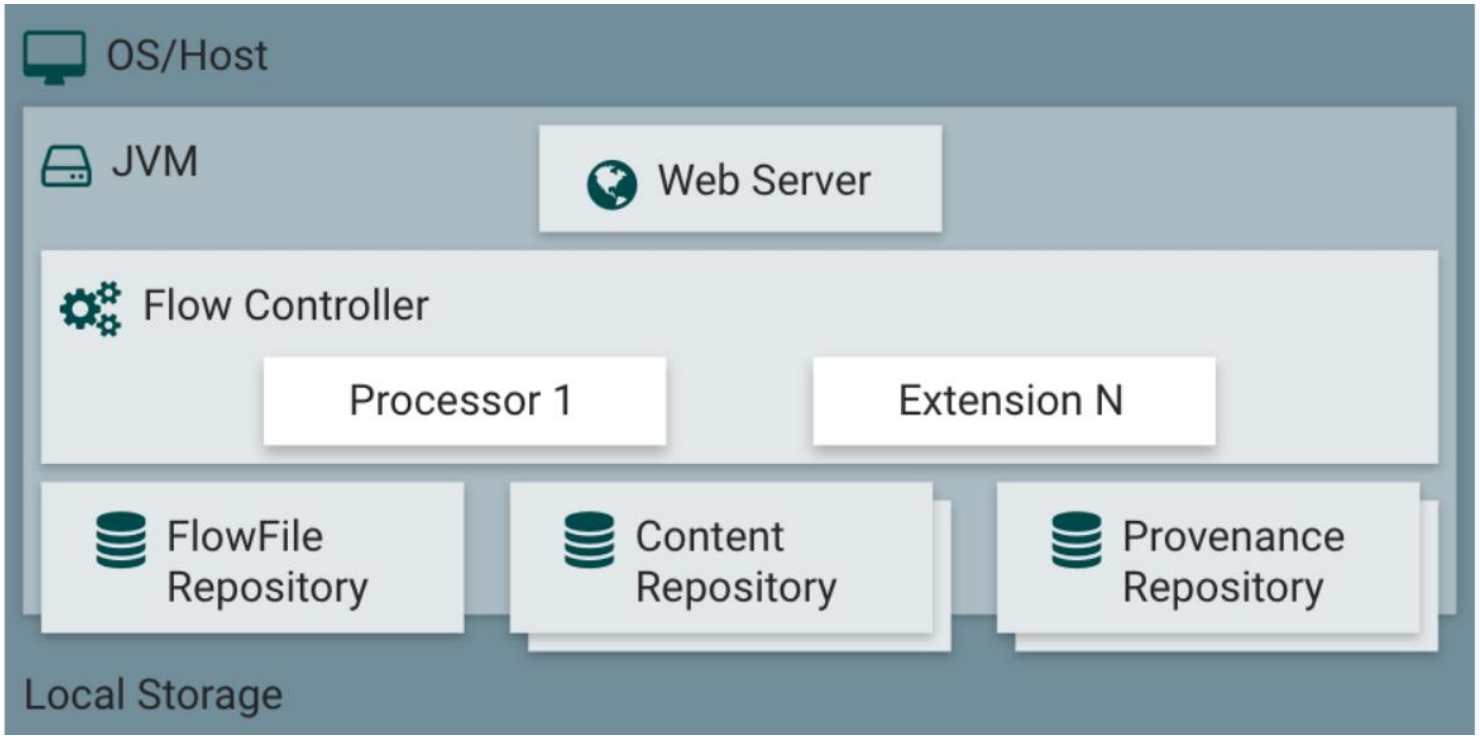
- Fiabilité :
  Bien que cela puisse varier selon les implémentations spécifiques, certains systèmes peuvent prendre en compte la capacité de traitement du nœud candidat, telle que la vitesse du CPU ou la quantité de RAM disponible, pour s'assurer qu'il peut gérer efficacement les tâches qui lui seront attribuées en tant que nœud primaire.

Composants et fonctionnalités

Les composants principaux de NiFi sur la JVM sont les suivants :

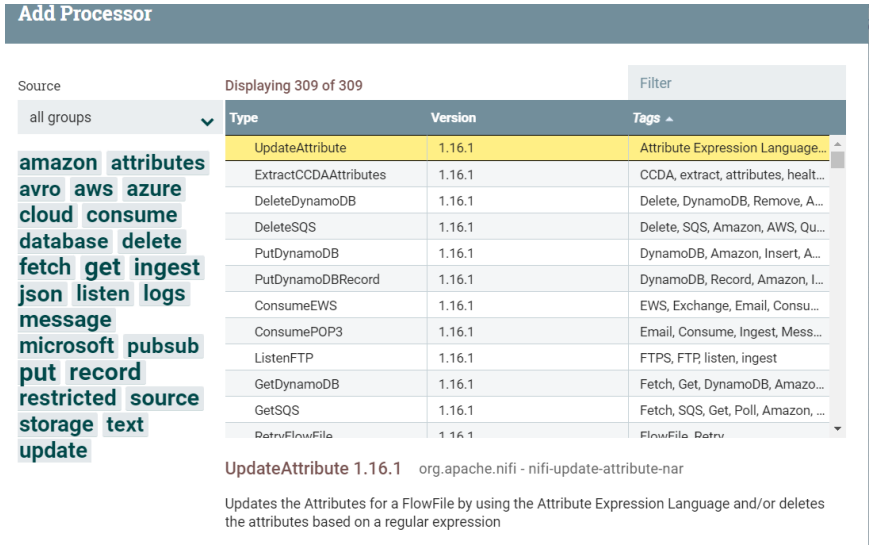
- **Serveur Web** qui héberge l'interface graphique,
- **Flow contrôler** = Orchestre les opérations c'est à dire fournir des tâches à exécuter aux **extensions**(processor, contrôle service, reporting task) et gère leur ordonnancement
  - Il n'y a qu'un seul Flow Controller par instance NiFi, mais ces Flow Controllers travaillent ensemble de manière distribuée et coordonnée via le Coordinator Node
  - Le Flow Controller sur chaque nœud gère l'exécution des tâches locales (traitement des flux de données) en suivant les directives du Coordinator Node,
  - Les Flow Controllers utilisent ZooKeeper pour synchroniser les informations sur les tâches et les répartitions. Ils conservent également des états internes pour le suivi des tâches
- **FlowFile Repository** dans lequel NiFi enregistre l'état d'un FlowFile (metadata)
- **Content Repository** où les données d'un FlowFile sont stockées (data)
- **Provenance Repository** où toutes les données d'événement de provenance sont stockées (historique des modifications)

Composants et fonctionnalités



Composants et fonctionnalités

En outre, NiFi possède plus de 200 connecteurs ou processeurs qui permettent de collecter en temps réel des données issues de plus de 100 sources : bases de données, messages, fichiers, flux Twitter, etc



Data Ingestion : des processeurs permettent de récupérer des données issues de différentes sources (GetTwitter, GetMongo, GetFile, GetHttp, etc.)

Data Transformation : compresser ou décompresser des données, convertir un format de données vers un autre (Xml, Json, Txt to Parquet, Avro, etc.) ou bien faire de la crypto et de l'encodage (ConvertCharacterSet, EncryptContent, etc.),

Les accès aux bases de données : par exemple le processeur Database Access ExecuteSQL ou bien PutHiveQl,

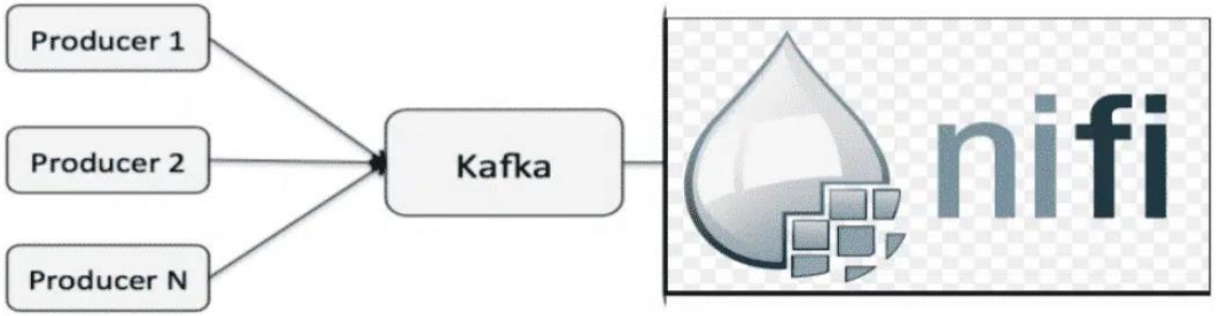
Data Egress/Sending Data : des processeurs pour envoyer des données vers des systèmes cible (PutSQL, PutKafka, PutMongo, etc.).

.

Composants et fonctionnalités

De nombreux cas d'usage sont réalisables avec ces deux superbes outils(nifi et kafka), comme construire un CDC (Change Data Capture), faire du streaming avec Kafka ou encore récupérer des logs de serveurs webs, etc

Apache NiFi en producteur de données pour Kafka



Apache NiFi en consommateur de données

Composants et fonctionnalités

NiFi tourne dans une JVM en mode local sur le système d'exploitation hôte (Windows, Linux ou Mac) mais on peut aussi déployer Apache NiFi en mode cluster, un de ses grands atouts une fois en production.

- Flowfile
- Processor
- flow
- Processor group(port input et output)
- Connection (directe entre Processor, par funnel , port input et output)
- RPG(remot process group)
- Contrôle service
- Template
- flow.xml.gz
- ……..

Exo1

1. Créer un Process group de nom SDA_2024

2. Générer des données au format suivant avec le processeur **GenerateFlowFile** :

```
{
  "destination": "local",
 "directory": "/tmp",
 "nomclient": "DIOP",
 "prix": 100,
 "ville": "Paris"
}
```

3. Utiliser le processeur **JoltTransformJSON** pour ajouter un timestamp au format yyyy-MM-dd_HHmmSS

```
[{
    "operation": "default",
    "spec": {
      "timestamp": "${now():format('yyyy-MM-dd_HHmmss')}"
    }
}

]
```

Sortie


```
{
  "destination" : "local",
  "directory" : "/tmp",
  "nomclient" : "DIOP",
  "prix" : 100,
  "ville" : "Paris",
  "timestamp" : "2024-07-01_082637"
}
```

Exo1

4. Utiliser le processeur **EvaluateJsonPath** afin de mettre la valeur de destination dans une variable NiFi nommée **destination** et la valeur de directory dans une variable NiFi nommée **rep**

5. Utiliser le processeur **RouteOnAttribute** pour router les données dans un process group nommé **SAVE_LOCAL** si destination=**local** Sinon, si destination=**elastic**, router les données vers un process group nommé **SAVE_ELASTIC**. Si aucune de ces conditions n'est remplie, les données ne sont pas routées.

6. Le nom du fichier déposé en local sera horodaté comme suit : **local_yyyy-MM-dd_HHmmss.json** et déposez-le dans le répertoire récupéré depuis le JSON (directory)

7. Pour la partie SAVE_ELASTIC, vous devez coder un processeur qui prend en entrée des noms de colonnes séparés par des virgules et crée un JSON qui ne contiendra que ces colonnes.

Par exemple, si votre JSON de base est :
{"destination":"elastic","directory":"/tmp","nomclient":"DIOP","prix":100,"ville":"Paris","timestamp":"2024-07-01_082637"}
et que vous donnez à votre processeur les colonnes nomclient et timestamp, en sortie vous obtiendrez :
{"nomclient":"DIOP", "timestamp":"2024-07-01_082637"}

- Votre processeur doit être capable de lever des exceptions en cas d'erreur, par exemple si les données d'entrée ne sont pas des JSON ou si une colonne donnée n'existe pas. Voir la page suivante pour la procédure de création

- Utilisez la configuration suivante pour insérer vos données dans Elasticsearch (index sda) et, pour vérifier si vos données y sont bien présentes, lancez la requête suivante

http://clustersdaelatsic.eastus.cloudapp.azure.com:9200/sda/_search?q=nomclient:VOTRENOM

Processus création processor

1. Sur la base du code(https://github.com/idiattara/Spark_DIATTARA/blob/main/NIFI.zip) fourni lors de la séance d'installation, créez un package nommé **jsonextractcopyrightdiattara** dans DHR-API-CLIENT-master\custom-processors\nifi-processor-processors\src\main\java\com\nifi.

2. Dans ce package, créez les deux classes suivantes : **FieldNotFoundException** et **JsonFieldExtractor**. Le contenu de ces classes se trouve aux adresses suivantes :https://github.com/idiattara/Spark_DIATTARA/blob/main/FieldNotFoundException.java
   https://github.com/idiattara/Spark_DIATTARA/blob/main/JsonFieldExtractor.java

3. Compilez la classe JsonFieldExtractor. Si le code est correct, commentez votre Main.

4. Créez un package nommé processor dans le package jsonextractcopyrightdiattara, puis créez dans ce package une classe **JsonFieldExtractorCustomProcessor** dont le contenu se trouve à l'adresse suivante :https://github.com/idiattara/Spark_DIATTARA/blob/main/JsonFieldExtractorCustomProcessor.java

5. Ajoutez **com.nifi.jsonextractcopyrightdiattara.processor.JsonFieldExtractorCustomProcessor** dans le fichier DHR-API-CLIENT-master\custom-processors\nifi-processor-processors\src\main\resources\META-INF\services**org.apache.nifi.processor.Processor.**

6. Maintenant, effectuez les tests unitaires en créant un package **jsonextractcopyrightdiattara** dans : DHR-API-CLIENT-master\custom-processors\nifi-processor-processors\src\test\java.

7. Créez la classe **JsonFieldExtractorCustomProcessorTest** dont le contenu se trouve à l'adresse suivant: https://github.com/idiattara/Spark_DIATTARA/blob/main/JsonFieldExtractorCustomProcessorTest.java

8. Maintenant, effectuez un clean puis un package, puis copiez dans le répertoire lib de NiFi le fichier DHR-API-CLIENT-master\custom-processors\nifi-processor-nar\target\nifi-processor-nar-1.0-SNAPSHOT.nar.

9. Enfin, redémarrez votre NiFi.

Exo1_BUS

1. Au lieu d’utiliser un GenerateFlowFile, mettez en place un service Web en utilisant le processeur HandleHttpRequest avec le chemin sdaweb. Si les données sont bien insérées dans la destination, renvoyez la réponse suivante : "Traitement effectué avec succès" avec un code de statut 200. Sinon, envoyez un code 500 avec la réponse "Traitement échoué "

2. Utilisez NiFi pour effectuer vos tests d’intégration ainsi que Python ou https://reqbin.com/curl

EXO2

1. Créer un processeur GenerateFlowFile
2. Mettre le contenu du processor dans un topic Kafka
3. Lire les données avec un script Python

EXO3

1. Utiliser Python pour envoyer un message dans Kafka

2. Lire les données avec NIFI

Exo4

Les Ops de l'entreprise Skill Data souhaitent orchestrer des jobs. Ils ont des scripts Python qui envoient des instructions sur un topic Kafka pour récupérer des données de MySQL et les mettre dans ElasticSearch.
Exemple d'instruction

```
{
    "requetes": [
                    {
                     "requete": "select  * from client",
                      "tablename":"client"
                     },
                    {
                   "requete": "select * from fourniseur",
                   "tablename":"fourniseur"
                    }
                ]
}
```

- Dans Elastic le nom de l' index sera le nom de la table, pensez à créer l'index template

- Il faut rajouter agent_timestamp sur chaque record de la donnée brutes

https://dbschema.com/jdbc-drivers/MySqlJdbcDriver.zip

jdbc:mysql://ip:port/databasename?characterEncoding=latin1&useConfigs=maxPerformance