TP : Kafka

L'objectif de ce TP est de manipuler des flux de données avec Kafka

Exercice 1 : Découverte de l'interface Kafka

1. Lancer un service kafka et jupyter-pyspark sur Onyxia. Ouvrir un terminal dans le service jupyter puis lancer la commande suivante pour installer l'interface utilisateur de Kafka

```
kubectl run kafka-ui --image provectuslabs/kafka-ui --env
DYNAMIC_CONFIG_ENABLED=true --port 8080 && kubectl expose pod/kafka-ui &&
kubectl create ingress --class onyxia --rule ${KUBERNETES_NAMESPACE}-
kafka-ui.user.lab.sspcloud.fr/*=kafka-ui:8080 kafka-ui
```

> N'oublier pas remplacer `<user>` par votre nom d'utilisateur

2. Connecter votre cluster Kafka en renseignant comme serveur `kafka` et le port `9092`
3. Depuis l'interface, créer un topic nommé `test` avec deux partitions
4. Créer un message `test1` et `test2` avec pour valeurs `1` dans le topic `test` dans chacune des deux partitions

Exercice 2 : Création d'un producer et d'un consumer Kafka en Python

1. Utiliser la bibliothèque `kafka-python-ng` pour créer un *producer* qui envoi les 3 premières lignes du csv `usagers-2022.csv` sous forme de 3 messages au topic `test`. Il faudra envoyer comme clé l'identifiant de l'accident et comme valeur sa gravité.
2. Créer ensuite un consumer qui lit et affiche les nouveaux messages générés par le producer précédent en commençant par les plus anciens
3. Ajouter ensuite de nouvelles lignes du csv dans le topic Kafka et écrire ces lignes dans une collection `usagers-reels` dans MongoDB

> La documentation est disponible ici

Exercice 3 : Utiliser Kafka Connect

1. Installer Kafka Connect avec les commandes suivantes :

```
kubectl run kafka-connect --image=titigmr/cp-kafka-connect:latest --
env=CONNECT_BOOTSTRAP_SERVERS=kafka:9092 --port=8083 --
env=CONNECT_REST_PORT=8082 --env=CONNECT_GROUP_ID="connect" --
env=CONNECT_CONFIG_STORAGE_TOPIC="connect-config" --
env=CONNECT_OFFSET_STORAGE_TOPIC="connect-offsets" --
env=CONNECT_STATUS_STORAGE_TOPIC="connect-status" --
```

```
env=CONNECT_KEY_CONVERTER="org.apache.kafka.connect.json.JsonConverter" --
env=CONNECT_VALUE_CONVERTER="org.apache.kafka.connect.json.JsonConverter"
--
env=CONNECT_INTERNAL_KEY_CONVERTER="org.apache.kafka.connect.json.JsonConv
erter" --
env=CONNECT_INTERNAL_VALUE_CONVERTER="org.apache.kafka.connect.json.JsonCo
nverter" --env=CONNECT_REST_ADVERTISED_HOST_NAME="kafka-connect" --
env=CONNECT_PLUGIN_PATH=/usr/share/java,/usr/share/confluent-hub-
components
```

```
kubectl expose pod/kakfa-connect
```

2. Dans l'interface utilisateur de Kafka, configurer Kafka-Connect en spécifiant comme serveur : `http://kafka-connect:8083`

3. Kafka Connect possède une API qui permet de le configurer. Lister les connecteurs disponible avec la commande suivante et vérifier que le connecteur MongoDB est présent.

```
curl http://kafka-connect:8083/connector-plugins/
```

> Le schéma de l'API est disponible ici

4. Lancer un connecteur avec comme nom `mongodb` qui synchronise les nouvelles données de la collection `test`. Utiliser la configuration suivante :

> La documentation est disponible ici

```
{
    "connector.class": "com.mongodb.kafka.connect.MongoSourceConnector",
    "connection.uri": "mongodb://<username>:<password>@<host>:
<port>/defaultdb",
    "database": "defaultdb",
    "collection": "test",
    "pipeline": "[{\"$match\": {}}]",
    "output.schema.infer.value": "true"
}
```

5. Insérer un nouveau document `{"usager": 1, "grav": 2}` dans la collection `test` et vérifer que le topic Kafka a bien crée le message via l'interface utilisateur