TP Kafka

Pour apprendre à manipuler Kafka, il faut tout d'abord effectuer quelques prérequis sur AWS et Ambari :
A. Démarrer votre instance ec2
B. Connectez-vous sur votre instance et démarrer docker si ce n'est pas déjà fait (sudo systemctl docker start)
C. Lancer les deux containers de la sandbox AWS (docker start ID-CONTAINER)
D. Attendre au moins une dizaine de minutes le temps que l'instance AWS se lance
E. Se connecter sur l'interface Ambari (http://<DNS-PUBLIC-EC2>:8080 (login/mdp : raj_ops / raj_ops)
F. Se rendre dans Kafka => Onglet Config.
G. Modifier le paramètre «listeners » à «PLAINTEXT://:9092 »
H. Modifier le paramètre « port » (dans les paramètres avancés) à « 9092
I. En haut de la page, cliquer sur « Restart All » pour Kafka

Une fois l'ensemble de ces prérequis effectués, se connecter en ssh sur le serveur Edge de votre cluster HDP. Deux manières :
a. Soit depuis ssh et votre instance ec2 en entrant dans votre container docker (docker exec -it <ID-CONTAINER> /bin/bash
b. Soit depuis l'interface webssh : http://<DNS-PUBLIC-EC2>:4200

Une fois connecté, vous réaliserez les étapes suivantes :

1) Rendez-vous dans le dossier suivant :

```
cd /usr/hdp/current/kafka-broker/
```

C'est ici que se trouve tous les utilitaires Kafka.

2) Créez un nouveau topic avec la commande suivante :

```
./bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic sio-topic
```

3) Listez l'ensemble des topics présents dans Kafka avec la commande suivante :

```
./bin/kafka-topics.sh --list --zookeeper localhost:2181
```

4) Affichez le détail d'un topic avec la commande suivante :

```
./bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic sio-topic
```

5) Utilisez l'utilitaire permettant de produire des messages fournis par Kafka :

```
./bin/kafka-console-producer.sh --broker-list localhost:9092 --topic sio-topic
```

6) Produisez quelques messages

7) Ouvrez un nouveau terminal et connecter vous à nouveau en ssh sur le serveur Edge de votre cluster HDP.

8) Dans le dossier des utilitaires kafka, utilisez l'utilitaire permettant de consommer des messages fournis par Kafka :

```
./bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic sio-topic
```

9) Vérifiez que vous recevez bien les messages publiés par le producer.

10) Relancer un consumer mais cette fois-ci afin de récupérer l'ensemble des messages publiés depuis le début

```
./bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic sio-topic --from-beginning
```

11) Modifier le topic sio-topic pour qu'il ait deux partitions

```
./bin/kafka-topics.sh --zookeeper localhost:2181 --alter --topic sio-topic --partitions 2
```

12) Relancer deux nouveaux consumer appartenant au même groupe de consumer sur le topic sio-topic

```
./bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic sio-topic --consumer-property group.id=sio-teacher
```

13) Nous allons maintenant créer un script python permettant de créer un producer Kafka. Pour cela, il est important d'avoir installer pip préalablement sur le cluster. Pour cela il faut télécharger le fichier python « get-pip.py »

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

14) Puis simplement exécuter ce fichier avec python

```
python get-pip.py
```

15) Pour utiliser kafka depuis python, vous aurez besoin d'installer le package kafka-python

```
pip install kafka-python
```

16) Créez un script python pour envoyer des données dans le producer sio-topic. Vous pourrez envoyer une données différentes toutes les secondes par exemple. Pour cela, vous aurez besoin d'importer et d'utiliser la class KafkaProducer depuis le package kafka-python, ainsi que d'utiliser sa fonction send.

```
from kafka import KafkaProducer
producer = KafkaProducer(bootstrap_servers='HOST:PORT')
producer.send('topic-name', b'valeur à envoyer')

producer.flush()


doc : https://kafka-python.readthedocs.io/en/master/apidoc/KafkaProducer.html
aide usage : https://kafka-python.readthedocs.io/en/master/usage.html
```

17) Exécuter votre script, vérifiez que vous recevez bien des données dans un consumer Kafka

18) Créez un second script pour consommer un topic kafka depuis python. Pour cela, vous aurez besoin d'importer et d'utiliser la classe KafkaConsumer depuis le package kafka-python.

```
from kafka import KafkaConsumer
consumer = KafkaConsumer('TOPIC', bootstrap_servers='HOST:PORT', group_id='MYGROUP')

KafkaConsumer(auto_offset_reset='earliest', enable_auto_commit=False)


doc : https://kafka-python.readthedocs.io/en/master/apidoc/KafkaConsumer.html
aide usage : https://kafka-python.readthedocs.io/en/master/usage.html
```

19) Insérer en continu des données au format json via un producer et dans un consumer, faites en sorte de sauvegarder les messages reçus dans un fichier csv sur hdfs toutes les N entrées. Pour sauvegarder sur hdfs depuis Python, vous aurez besoin du package hdfs et des classes et fonctions suivantes :

```
from hdfs import InsecureClient
client = InsecureClient('localhost:50070')
client.write('/PATH/TO/FILE/FILE.CSV', data=DATA, encoding='utf-8')

doc : https://hdfscli.readthedocs.io/en/latest/api.html
```

20) Créez un producer qui récupère les données de l'API transilien régulièrement pour une gare donnée. Passez les nouvelles données récupérées dans un topic kafka que vous pourrez par la suite consommer. Sauvegarder les données