TP4 Kafka et Nifi

Pas besoin de VM pour ce cours, on va travailler directement depuis vos poste telecom. Le TP a été testé depuis une VM debian.

Exercice 1: Kafka

Kafka est un Message Oriented Middleware. Nous allons l'installer et le tester. Nous testerons les api entrantes et sortantes, mais aussi la robustesse de Kafka, en killant certains brokers Kafka pendant les écritures ou la lecture de messages.

a. Installation

Dans un premier temps, nous allons procéder à l'installation d'un broker Kafka pour comprendre les notions de consumer et producer. Ensuite nous déploierons un cluster de 3 brokers Kafka sur lesquels nous testerons la robustesse.
Pour rappel, nous aurons besoin d'un service Zookeeper pour utiliser le service Kafka. Nous pourrions utiliser le zookeeper présent sur les VMs cloudera. pour des raisons de performance, je conseille d'utiliser directement les machines centos de l'école sur lesquelles nous lancerons Zookeeper et Kafka.

Créer un dossier "kafka" sur le filesystem Linux.
Récupérer le package kafka:
wget **https://www-eu.apache.org/dist/kafka/2.3.0/kafka_2.12-2.3.0.tgz**
Décompresser le package :
tar -xzf kafka_2.11-2.0.0.tgz
Entrer dans le répertoire :
cd kafka_2.11-2.0.0
Démarrez le service Zookeeper embarqué :
bin/zookeeper-server-start.sh config/zookeeper.properties

Vous pouvez vérifier les Znodes présents sur Zookeeper avec la cli (ouvrez un autre terminal):
bin/zookeeper-shell.sh localhost:2181
ls /

Une fois que Zookeeper a démarré, ouvrez un autre terminal et lancez le serveur Kafka :
bin/kafka-server-start.sh config/server.properties
Notez que cette commande prend en entrée un fichier .properties détaillant la configuration du broker Kafka qui va être lancé. Notez la propriété broker.id, permettant d'identifier les brokers.
Parmis les options intéressantes, notez aussi le nombre de partitions par topic par défaut, la durée de rétention des messages dans Kafka par défaut. Ces configurations sont spécifiques aux topics.

Vérifiez à nouveau les Znodes créés: ls /

b. Création de topic

Le service devrait maintenant tourner. Testez le en créant un premier topic Kafka :
bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic fr.telecom.tpkafka.kafkatopictest
Le message “Created topic "fr.telecom.tpkafka.kafkatopictest".” devrait s’afficher dans les logs du broker. Vous pouvez aussi utiliser la commande ci-dessous pour lister les topics déclarés sur votre cluster Kafka :
bin/kafka-topics.sh --list --zookeeper localhost:2181
Vous retrouvez également ces informations via la cli Zookeeper, dans les nouveaux Znodes créés: ls /brokers/topics

c. producer Kafka

A présent, testez le producer shell de kafka pour écrire des messages dans le topic :
bin/kafka-console-producer.sh --broker-list localhost:9092 --topic fr.telecom.tpkafka.kafkatopictest
Cette commande vous ouvre une session d’écriture vers votre topic Kafka. Ecrivez un premier message, faites entrée pour envoyer le message au topic.

d. consumer Kafka

Dans un autre terminal, lancez un consumer via :
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic fr.telecom.tpkafka.kafkatopictest --from-beginning
Vous devriez voir apparaître les messages que vous avez envoyé depuis votre producer.
Votre consumer possède un offset sur le topic Kafka qui vous permet de récupérer les derniers messages non lus. Cependant, l’option --from-beginning vous permet de reparcourir l’ensemble des données disponibles dans le topic, sans se soucier de l’offset.
Ces deux commandes possèdent plusieurs options que vous pouvez afficher en les lançant sans arguments.
Tapez quelques messages dans votre producer et regardez les apparaitres dans le consumer.

e. Un cluster multi-broker

Pour l’instant nous n’avons utilisé qu’un seul broker. Pour tester le parallélisme et la haute tolérance aux pannes de Kafka nous allons créer un cluster de 3 brokers.
Un broker est définit par un fichier de configuration, dans lequel est spécifié l’id du broker, le port sur lequel le broker écoute, et le dossier dans lequel les données seront écrites.
Afin de spécifier les deux autres brokers, créez deux nouveaux fichiers properties en partant du fichier server.properties.
cp config/server.properties config/server-1.properties
cp config/server.properties config/server-2.properties
Dans ces deux fichiers, modifiez les propriétés énoncées plus haut afin d’avoir:

config/server-1.properties:

broker.id=1
listeners=PLAINTEXT://:9093
log.dir=/tmp/kafka-logs-1

config/server-2.properties:

broker.id=2
listeners=PLAINTEXT://:9094
log.dir=/tmp/kafka-logs-2

Un de nos brokers tourne déjà. Nous allons maintenant lancer les deux autres brokers.
bin/kafka-server-start.sh config/server-1.properties &
bin/kafka-server-start.sh config/server-2.properties &

Créez maintenant un nouveau topic, de 1 partition, avec un facteur de réplication de 3 :
bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 3 --partitions 1 --topic fr.telecom.tpkafka.topicreplication

Pour voir la façon dont sont réparties les partitions et ls réplicas sur un topic, tapez la commande ci-dessous :
bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic fr.telecom.tpkafka.topicreplication

largo@metroid:~/workspace/kafka/kafka_2.11-0.11.0.1$ bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic fr.telecom.tpkafka.topicreplication
Topic:fr.telecom.tpkafka.topicreplication PartitionCount:1 ReplicationFactor:3 Configs:

Topic: fr.telecom.tpkafka.topicreplication Partition: 0 Leader: 2 Replicas: 1,2,0 Isr: 1,2,0

Dans mon exemple, ci dessus, on voit que le leader est le broker numéro 2, et qu'il y a 3 réplicas (sur les brokers 1,2 et 0).
Nous allons maintenant tester la tolérance aux pannes.
Ecrivez quelques messages dans le topic via la commande du producer.
bin/kafka-console-producer.sh --broker-list localhost:9092,localhost:9093,localhost:9094 --topic fr.telecom.tpkafka.topicreplication

Consommez les avec bin/kafka-console-consumer.sh, comme précédemment.

Quittez le consumer

Maintenant Killez le broker sur lequel est la partition leader.
(vous pouvez quitter le terminal du broker correspondant, ou utiliser ps faux | grep kafka pour retrouver l'id puis killer le process avec kill -9)
Exemple pour le broker 2 :
largo@metroid:~/workspace/kafka/kafka_2.11-0.11.0.1$ ps faux | grep server-2.properties
largo 13440 1.5 1.9 4660144 315396 pts/4 Sl 06:36

La commande sera :
kill -9 13440
Via la commande describe, notez que le leader a changé.
Avec le producer, envoyez quelques nouveaux messages.
Relancez le consumer. Observez que les données sont toujours accessibles.

Relancez maintenant le broker que vous avez éteint.

3. Créez un script bash qui envoie un message contenant la date et l'heure à la nanoseconde tout les secondes dans le topic.
Utilisez les commandes unix suivantes:

date
while
sleep

Pour écrire un message dans un topic programmatiquement:

echo " ... " | bin/kafka-console-producer.sh --broker-list localhost:9092,localhost:9093,localhost:9094 --topic fr.telecom.tpkafka.topicreplication

Exécutez ce script, observez dans le consumer les données arriver.

Exercice 2 : Nifi

On va installer nifi et tenter de l'interfacer avec kafka:
Ouvrez un nouveau terminal
Créez un dossier nifi dans /tmp (a cause de votre quota sur vos sessions)
Attention, le dossier tmp ne sera pas sauvegardé dans votre session.
cd /tmp
mkdir nifi
cd nifi
récupérez nifi:
wget **https://www-eu.apache.org/dist/nifi/1.9.2/nifi-1.9.2-bin.tar.gz**
Décompressez le
tar -xzf nifi-1.7.1-bin.tar.gz
cd nifi-1.7.1

lancez nifi
bin/nifi.sh start

Rendez vous sur:

http://localhost:8080/nifi
Le lancement de nifi peut prendre quelques minutes.

https://nifi.apache.org/docs.html

nifi fonctionne beaucoup avec le glissé déposé. Trouvez l'icône processor en haut a gauche, et glissez déposez la sur l'écran principal.
Observez les différents processors accessibles, HDFS HBase, AWS, etc…

1: On va consommer notre topic kafka et écrire les messages dans des fichiers sur le fs local.
Utilisez pour cela les processors KafkaConsume et PutFile.

Pour configurer les processors, clic droit, configure. Ensuite on modifie les properties. Ne fonctionne que si le processor ne tourne pas.

KafkaConsume :
Consommez le topic fr.telecom.tpkafka.topicreplication, configurez bien les 3 brokers. Le groupid est telecom (ou peut importe dans notre cas).

PutFile:
Ecrivez dans le dossier ~/nifi/output1

N'oubliez pas que toute les connexions produites par les processors doivent être connectées à quelque chose ou auto-terminées.

Une fois le flot représenté, lancez le.
Observez ~/nifi/output1

2: On va maintenant merger les messages avant de les ecrire dans ~/nifi/output2.
utilisez le processor mergeContent.
Ecrivez des fichiers de 1ko minimum.
indice: on veut séparer les messages par un retour à la ligne vu que c'est du texte. Pour cela, la delimiter strategy sera text, et le demarcator un retour à la ligne (shift+enter)
Pas besoin d'un autre consumer Kafka.

Une fois le tp terminé, n'oubliez pas d'éteindre tous les process en cours, nifi, kafka, bash.

Solutions

Script:

```
#!/bin/bash
while sleep 1
do
    echo "message ecrit le $(date +%H:%M:%S:%N)" | bin/kafka-console-producer.sh
--broker-list localhost:9092,localhost:9093,localhost:9094 --topic
fr.telecom.tpkafka.topicreplication
done
```