TP 1 - Kafka Installation et commande de base

Objectif : Découvrir Apache Kafka, installer l'environnement et manipuler les concepts de base (topics, producers, consumers).

Prérequis : - Java JDK 17+ installé - Docker et Docker Compose installés

---

Partie 1 - Installation du JAR

1.1 Téléchargement de Kafka

1. Rendez-vous sur le site officiel : https://kafka.apache.org/downloads
2. Téléchargez la dernière version stable
3. Décompressez l'archive dans un répertoire de travail

1.2 Démarrage de l'environnement Kafka

Kafka nécessite un broker, par défaut ZooKeeper est recommandé. Néanmoins, depuis les dernières version et pour travailler en local, il existe une version KRaft. Pour cette partie, nous utiliserons cette version

1. **Générer un ID de cluster** :

```
KAFKA_CLUSTER_ID="$(bin/kafka-storage.sh random-uuid)"
```

2. **Formater le répertoire de logs** :

```
bin/kafka-storage.sh format --standalone -t $KAFKA_CLUSTER_ID -c config/server.properties
```

3. **Démarrer le serveur Kafka** :

```
bin/kafka-server-start.sh config/server.properties
```

4. Vérifiez que Kafka est démarré (le terminal doit afficher des logs sans erreurs)

Laissez ce terminal ouvert - Kafka tourne en foreground

1.3 Manipulation des Topics

Ouvrez un **nouveau terminal** dans le même répertoire Kafka.

1. **Créer un topic** nommé `test-topic` avec 3 partitions en utilisant la commande bin/kafka-topics.sh
2. **Lister les topics**
3. **Décrire le topic** créé précédemment `test-topic`

1.4 Producteur et Consommateur en ligne de commande

Ouvrez un **nouveau terminal** dans le même répertoire Kafka.

1. **Démarrer un producteur** sur le topic `test-topic`

Tapez quelques messages. Chaque ligne correspond à un message.

Ouvrez un **nouveau terminal** dans le même répertoire Kafka.

2. **Démarrer un consommateur** sur le topic `test-topic`

- Pourquoi les messages envoyez précédemment ne sont-ils pas reçu ?

3. **Expérimentez** :
    - Envoyez de nouveaux messages depuis le producteur
    - Observez-les apparaître dans le consommateur
    - Arrêtez le consommateur (Ctrl+C) et démarrez-le **avec** `--from-beginning`. Que se passe-t-il ?
4. Rajoutez un **consumer group** lors du lancement du topic `avec --group group`
5. **Questions à répondre** :
    - Lancez ce consommateur 2 fois en parallèle (2 terminaux différents avec même groupe). Envoyez des messages. Comment sont-ils distribués ?

1.5 Arrêt de l'installation JAR

Arrêtez tous les processus Kafka lancés précédemment.

Partie 2 - Utilisation avec Docker

Docker simplifie grandement le déploiement de Kafka. Nous allons recréer notre environnement avec Docker Compose.

2.1 Configuration Docker Compose

1. **Créer un fichier `docker-compose-zookeeper.yml` en comprenant l'image `confluentinc/cp-kafka:7.4.4` avec un ZooKeeper `confluentinc/cp-zookeeper:7.4.4`**
2. **Créer un fichier `docker-compose-kraft.yml` en comprenant l'image `confluentinc/cp-kafka:8.0.3` mode KRaft**

Note: Pour rajouter un visualiseur :

```yaml
services:
  kafka-ui:
    image: kafbat/kafka-ui:latest
```

```
    container_name: kafka-ui
    depends_on:
      - kafka-broker
    ports:
      - "9091:8080"
    environment:
      DYNAMIC_CONFIG_ENABLED: true
      KAFKA_CLUSTERS_0_NAME: kafka-broker
      KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS: kafka-broker:29092
    networks:
      - network
    restart: unless-stopped
```

3. **Démarrer Kafka** :

```
docker compose -f docker-compose-kraft up -d
```

4. **Vérifier les logs** :

```
docker compose logs -f kafka-broker
```

Attendez que Kafka soit complètement démarré. Un message devrait appararaitre dans les logs :

```
INFO [KafkaRaftServer nodeId=1] Kafka Server started (kafka.server.KafkaRaftServer)
```

2.2 Manipulation des Topics avec Docker

1. **Connexion au docker** :

```
docker exec -it broker sh
```

2. **Créer un topic `etudiants`**
3. **Lister les topics et décrire le topic `etudiants`**
4. **Créer un producteur Docker sur le topic `etudiants`**

Envoyez quelques messages représentant des étudiants. (Attention : Une ligne correspond à un message)

```
{
    "firstName": "test",
    "lastName": "test",
    "age": 21,
    "engineeringDegree": "IT"
}
```

5. **Créer un consumateur Docker sur le topic `etudiants`**

2.3 Nettoyage

```
docker compose down -v
```

Questions de synthèse

1. Quelle est la différence entre une partition et un topic ?
2. À quoi sert un consumer group ?
3. Quels sont les avantages de Docker pour le développement avec Kafka ?
4. Que se passe-t-il si on a plus de consommateurs que de partitions dans un groupe ?
5. Comment Kafka garantit-il l'ordre des messages ?