TP6- Mise en place de Kafka

Découvrir Apache Kafka et manipuler les concepts fondamentaux via la ligne de commande.

Objectifs : A la fin de ce guide, vous serez capable de :

- Installer et démarrer Kafka avec Docker Compose (Broker + Zookeeper)
- Créer et gérer des Topics (partitions, réplication, configuration)
- Publier des événements avec kafka-console-producer
- Consommer des événements avec kafka-console-consumer et Consumer Groups
- Observer le comportement distribué (partitions, offsets, parallélisme)
- Manipuler les offsets pour rejouer des messages (rejouabilité)

Partie 1 : Installation et configuration de KAFKA

Étape 1 : Télécharger le fichier docker-compose.yml

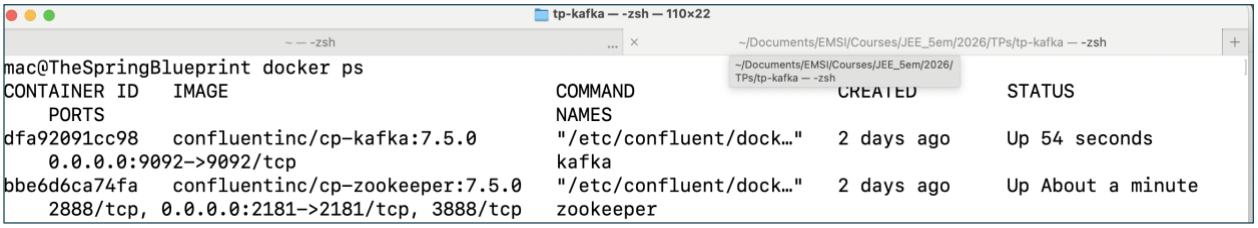
Étape 2 : Lancez les conteneurs en mode détaché :

```
$ docker-compose up -d
```

Cette commande démarre les services définis dans le fichier docker-compose.yml en arrière-plan (deamon).

Étape 3 : Vérifier si les 2 conteneurs sont en cours d'exécution.

```
$ docker-compose ps
```



Étape 4 : Accéder au conteneur Kafka

```
$ docker exec -it kafka bash
```

Partie 2 : Création et gestion des Topics

Étape 1 : Lister les topics existants

*Résultat attendu : Aucun topic

Étape 2 : Créer votre premier topic

```
[appuser@dfa92091cc98 ~]$ kafka-topics --bootstrap-server localhost:9092 \
>   --create \
>   --topic doctorat-events \
>   --partitions 3 \
>   --replication-factor 1
Created topic doctorat-events.
```

Argument	Description
`--bootstrap-server localhost:9092`	Adresse du broker Kafka auquel se connecter.
`--create`	Indique que l'on veut créer un nouveau topic.
`--topic doctorat-events`	Nom du topic à créer.
`--partitions 3`	Définit que le topic aura 3 partitions.
`--replication-factor 1`	Définit que chaque partition possède une seule réplique.

Étape 3 : Vérifier les détails du topic

```
[appuser@dfa92091cc98 ~]$ kafka-topics --bootstrap-server localhost:9092   --describe   --topic doctorat-events
Topic: doctorat-events  TopicId: CpfcS_JDTti-QRRIHBbrEQ PartitionCount: 3       ReplicationFactor: 1    Configs:
        Topic: doctorat-events  Partition: 0    Leader: 1       Replicas: 1     Isr: 1
        Topic: doctorat-events  Partition: 1    Leader: 1       Replicas: 1     Isr: 1
        Topic: doctorat-events  Partition: 2    Leader: 1       Replicas: 1     Isr: 1
```

Étape 4 : Créer un topic avec configuration avancée

```
[appuser@dfa92091cc98 ~]$ kafka-topics --bootstrap-server localhost:9092 \
>   --create \
>   --topic notification-events \
>   --partitions 2 \
>   --replication-factor 1 \
>   --config retention.ms=3600000 \
>   --config compression.type=snappy
Created topic notification-events.
```

Argument	Description
--config retention.ms=3600000	Définit la durée de rétention des messages du topic à 3 600 000 ms (1 heure).
--config compression.type=snappy	Configure Kafka pour compresser les messages du topic avec l'algorithme Snappy.

Étape 5 : Modifier la configuration d'un topic existant

```
[appuser@dfa92091cc98 ~]$ kafka-configs --bootstrap-server localhost:9092 \
>   --alter \
>   --entity-type topics \
>   --entity-name doctorat-events \
>   --add-config retention.ms=604800000
Completed updating config for topic doctorat-events.
```

- **Vérifiez la modification :**

```
[appuser@dfa92091cc98 ~]$ kafka-configs --bootstrap-server localhost:9092 \
>   --describe \
>   --entity-type topics \
>   --entity-name doctorat-events
Dynamic configs for topic doctorat-events are:
  retention.ms=604800000 sensitive=false synonyms={DYNAMIC_TOPIC_CONFIG:retention.ms=604800000}
```

Étape 6 : Supprimer un topic

```
[appuser@dfa92091cc98 ~]$ kafka-topics --bootstrap-server localhost:9092 \
>   --delete \
>   --topic notification-events
[appuser@dfa92091cc98 ~]$ kafka-topics --bootstrap-server localhost:9092 --list
doctorat-events
```

Exercice :

Créez un topic pour votre projet avec ces caractéristiques :

- Topic name : soutenance-events
- Partitions : 5
- Rétention : 30 jours
- Compression : lz4

Partie 3 : Publication et consommation de Messages

Étape 1 : Publier votre premier message

Ouvrez un producer en mode console interactif :

```
[appuser@dfa92091cc98 ~]$ kafka-console-producer --topic doctorat-events --bootstrap-server localhost:9092
>
```

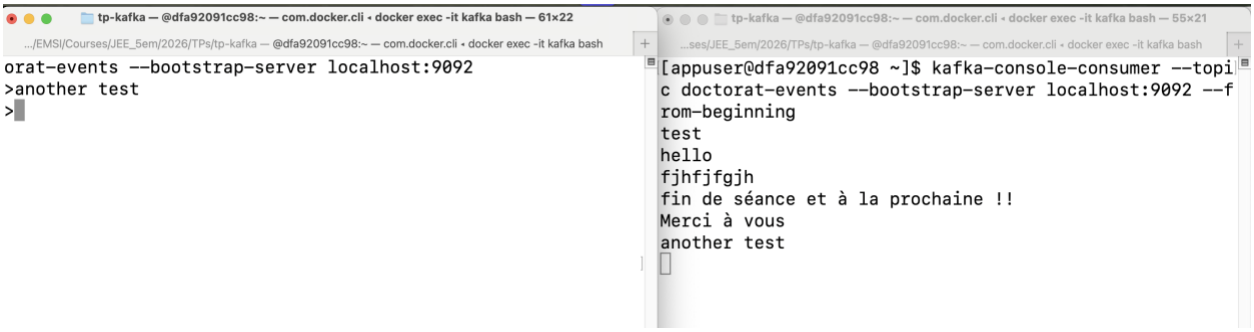
Pour quitter : Ctrl + C

Étape 2 : Consommer les messages

Dans un second terminal (ou nouvelle fenêtre) et accédez au conteneur et lancez un consumer pour lire les messages :

```
[appuser@dfa92091cc98 ~]$ kafka-console-consumer --topic doctorat-events --bootstrap-server localhost:9092
--from-beginning
```

Étape 3 : Tester la publication en temps réel



Étape 4 : Publier des messages au format JSON

```
[appuser@dfa92091cc98 ~]$ kafka-console-producer --bootstrap-
server localhost:9092 \
>   --topic doctorat-events \
>   --property "parse.key=true" \
>   --property "key.separator=:"
>DOC-001:{"event":"DoctorantInscrit","nom":"Ahmed","date":"20
25-12-02"}
>
```

Pourquoi on utilise les clés ? Messages avec même clé ➔ vont dans la même partition, cela garantit l'ordre des événement pour un même doctorant.

Étape 5 : Consommer avec affichage des clés

```
[appuser@dfa92091cc98 ~]$ kafka-console-consumer --bootstrap-server localhost:9092 \
>   --topic doctorat-events \
>   --from-beginning \
>   --property print.key=true \
>   --property key.separator=" => "
null => un autre message
null => test
null => hello
null => fjhfjfgjh
null => fin de séance et à la prochaine !!
null => Merci à vous
null => another test
DOC-001 => {"event":"DoctorantInscrit","nom":"Ahmed","date":"2025-12-02"}
```

Partie 4 : Consumer groups, partitions et rejouabilité des messages

Étape 1 : Vérification du topic et de ses partitions

Vérifiez que le topic doctorat-events existe avec 3 partitions

```
[appuser@dfa92091cc98 ~]$ kafka-topics --bootstrap-server localhost:9092 --describe --topic doctorat-events
Topic: doctorat-events  TopicId: CpfcS_JDTti-QRRIHBbrEQ PartitionCount: 3       ReplicationFactor: 1    Configs: retention.ms=604800000
        Topic: doctorat-events  Partition: 0    Leader: 1       Replicas: 1     Isr: 1
        Topic: doctorat-events  Partition: 1    Leader: 1       Replicas: 1     Isr: 1
        Topic: doctorat-events  Partition: 2    Leader: 1       Replicas: 1     Isr: 1
```

Étape 2 : Produire des messages de test

```
[appuser@dfa92091cc98 ~]$ kafka-console-producer --bootstrap-server localhost:9092 --topic doctorat-events
>Message 1
>Message 2
>Message 3
>Message 4
>Message 5
>Message 6
>Message 7
>Message 8
>Message 9
>Message 10
>Message 11
>Message 12
>Message 13
```

Étape 3 : Lancer le premier consumer avec un groupe

```
[appuser@dfa92091cc98 ~]$ kafka-console-consumer --bootstrap-server localhost:9092   --topic doctorat-events   --group doctorat-group
--from-beginning
```

Résultat attendu : Le consumer affiche les 12 messages (car il est seul dans le groupe, il consomme toutes les partitions).

```
Message 1
Message 2
Message 3
Message 4
Message 5
Message 6
Message 7
Message 8
Message 9
Message 10
Message 11
Message 12
Message 13
```

Étape 4 : Vérifier l'assignation des partitions au consumer

Dans un 3ème terminal :

```
[appuser@dfa92091cc98 ~]$ kafka-consumer-groups --bootstrap-server localhost:9092   --group doctorat-group   --describe

GROUP           TOPIC           PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG             CONSUMER-ID                                     HOST
doctorat-group  doctorat-events 0          0               0               0               console-consumer-59b23565-bee9-4a69-beb6-98aa6ea35674 /127.0.0.1
doctorat-group  doctorat-events 1          0               0               0               console-consumer-59b23565-bee9-4a69-beb6-98aa6ea35674 /127.0.0.1
doctorat-group  doctorat-events 2          13              13              0               console-consumer-59b23565-bee9-4a69-beb6-98aa6ea35674 /127.0.0.1
```

Résultat attendu : Affichage montrant que le Consumer 1 est assigné aux 3 partitions (0, 1, 2), avec les offsets actuels.

Étape 5 : Produire de nouveaux messages

Gardez le Consumer 1 actif. Dans le terminal du producer, envoyez 6 nouveaux messages.

Observation : Le Consumer 1 affiche immédiatement ces nouveaux messages en temps réel.

Étape 6 : Ajouter un deuxième consumer dans le même groupe

Sans arrêter le Consumer 1, ouvrez un quatrième terminal et lancez le Consumer 2 dans le même groupe :

```
[appuser@dfa92091cc98 ~]$  kafka-console-consumer --bootstrap-server localhost:9092   --topic doctorat-events   --group doctorat-group
```

Revenez à l'étape 4 pour vérifier l'assignation

```
[appuser@dfa92091cc98 ~]$ kafka-consumer-groups --bootstrap-server localhost:9092 \
>   --group doctorat-group \
>   --describe

GROUP           TOPIC           PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG             CONSUMER-ID                                     HOST            CLIENT-ID
doctorat-group  doctorat-events 0          0               0               0               console-consumer-4557a266-9158-493c-8995-d774c3c3b33b /127.0.0.1      console-consume
r
doctorat-group  doctorat-events 1          0               0               0               console-consumer-4557a266-9158-493c-8995-d774c3c3b33b /127.0.0.1      console-consume
r
doctorat-group  doctorat-events 2          19              19              0               console-consumer-59b23565-bee9-4a69-beb6-98aa6ea35674 /127.0.0.1      console-consume
r
```

Observations :

- Deux CONSUMER-ID différents
- Kafka rééquilibre les partitions entre Consumer 1 et Consumer 2
- Chaque consumer se voit assigner environ 1-2 partitions (ex: Consumer 1 → partition2, Consumer 2 → partition 0 et 1)

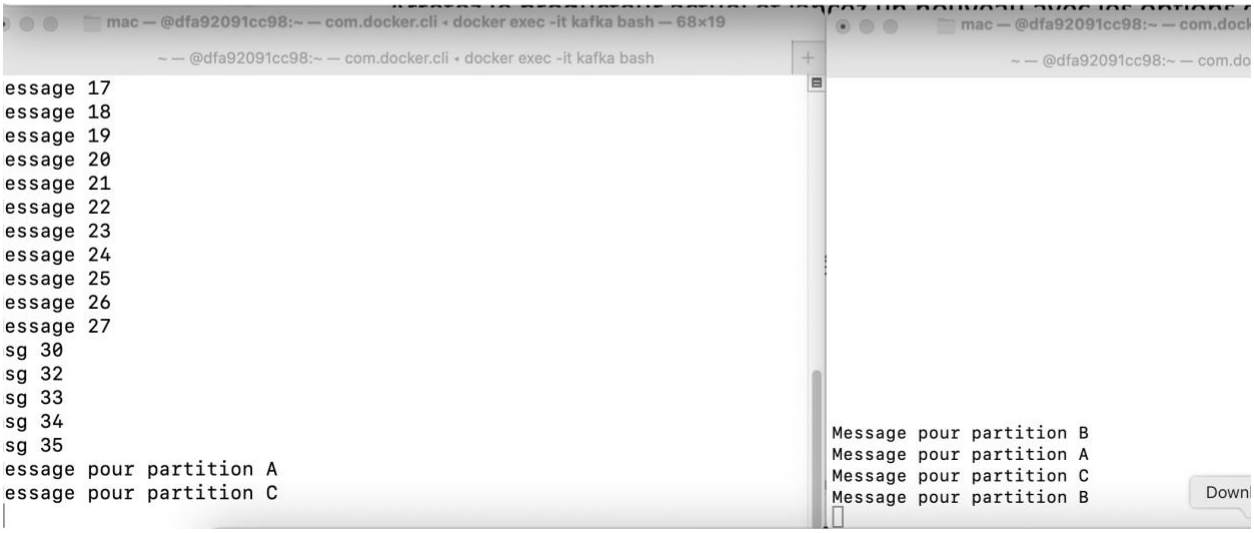
Étape 7 : Tester la répartition par partition

Arrêtez le producteur actuel et lancez un nouveau avec les options de saisi des clés

```
[appuser@dfa92091cc98 ~]$ kafka-console-producer --bootstrap-server localhost:9092 \
>   --topic doctorat-events \
>   --property "parse.key=true" \
>   --property "key.separator=:"
>
```

Envoyez des messages avec des clés différentes :

```
[appuser@dfa92091cc98 ~]$ kafka-console-producer --bootstrap-server localhost:9092 \
>   --topic doctorat-events \
>   --property "parse.key=true" \
>   --property "key.separator=:"
>id1:Message pour partition A
id2:Message pour partition B
id3:Message pour partition C
id4:Message pour partition A
id5:Message pour partition B
id6:Message pour partition C
>>>>>>
```



Observation : Regardez la partition des messages entre les consumers. Ajoutez + de consumer et observez !