6️ TP06 : Kafka et ksqlDB – Suivi des changements Netflix

1️ Objectif du TP

Le but de ce TP est de **surveiller les changements dans la production de séries et films Netflix** (par exemple, changement du nombre d'épisodes d'une saison).

On utilise **Kafka** pour gérer les flux de données et **ksqlDB** pour filtrer, transformer, enrichir et agréger ces flux en temps réel.

2️ Architecture générale

- **Topics Kafka** :
  - `titles` : contient les informations des titres (films, séries).
  - `production_changes` : contient les changements de production (budget, durée de saison, date de sortie, etc.).
- **ksqlDB** : consomme les topics, transforme et enrichit les données, puis crée des flux dérivés et des tables matérialisées.
- **Clients** :
  - **Push query** : reçoit les changements en continu.
  - **Pull query** : récupère l'état actuel des données à un instant donné.

3️ Étapes du TP

Étape 1 : Lancer l'environnement

- Lancer **Docker Compose** pour démarrer Zookeeper, Kafka, ksqlDB Server et ksqlDB CLI.

- Commandes principales :

```
docker-compose up -d
docker exec -it ksqldb-cli ksql http://ksqldb-server:8088
docker exec -it ksqldb-cli ksql http://ksqldb-server:8088/
```

Étape 2 : Créer le type personnalisé `season_length`

- Permet de représenter les objets `before` et `after` de `production_changes`.
- Commande :

```
CREATE TYPE season_length AS STRUCT<season_id INT, episode_count INT
>;
```

- Vérification : `SHOW TYPES;`
- Suppression si besoin : `DROP TYPE season_length;`

Étape 3 : Créer les collections (Streams)

- **Collection** `titles` : stream des titres avec `id` comme identifiant.
- **Collection** `production_changes` : stream des changements avec `rowkey` comme identifiant et `created_at` pour les timestamps.
- Exemple :

```
CREATE STREAM titles (
    id INT KEY,
    title STRING
) WITH (KAFKA_TOPIC='titles', VALUE_FORMAT='JSON', PARTITIONS=4);

CREATE STREAM production_changes (
    rowkey STRING KEY,
    title_id INT,
```

```
    change_type STRING,
    before season_length,
    after season_length,
    created_at TIMESTAMP
) WITH (KAFKA_TOPIC='production_changes', VALUE_FORMAT='JSON', TIME
STAMP='created_at', PARTITIONS=4);
```

Étape 4 : Insérer des données exemple

- Dans `titles` et `production_changes` :

```
INSERT INTO titles (id, title) VALUES (1, 'Stranger Things');
INSERT INTO production_changes (rowkey, title_id, change_type, before, afte
r, created_at)
VALUES ('key1', 1, 'season_length', STRUCT(1,12), STRUCT(1,8), '2021-02-08 1
1:30:00');
```

Étape 5 : Filtrer et transformer les données

- On ne conserve que les changements de type `season_length`.
- Création d'un **stream dérivé** `season_length_changes` :

```
CREATE STREAM season_length_changes AS
SELECT
    ROWKEY,
    title_id,
    created_at,
    COALESCE(after→season_id, before→season_id) AS season_id,
    before→episode_count AS old_episode_count,
    after→episode_count AS new_episode_count
FROM production_changes
WHERE change_type='season_length';
```

Étape 6 : Enrichissement des données

- Joindre `season_length_changes` avec `titles` pour ajouter le nom du titre.
- Création du stream enrichi `season_length_changes_enriched` :

```
CREATE STREAM season_length_changes_enriched AS
SELECT s.*, t.title
FROM season_length_changes s
LEFT JOIN titles t
ON s.title_id = t.id;
```

Étape 7 : Agrégation et tables matérialisées

- Créer une table `season_length_change_counts` pour compter le nombre de changements par titre et garder le dernier nombre d'épisodes.
- Utilisation de **fenêtres temporelles (TUMBLING)** d'une heure :

```
CREATE TABLE season_length_change_counts AS
SELECT
   title,
   LATEST_BY_OFFSET(new_episode_count) AS episode_count,
   COUNT(*) AS change_count
FROM season_length_changes_enriched
WINDOW TUMBLING (SIZE 1 HOUR)
GROUP BY title;
```

Étape 8 : Requêtes Push et Pull

- **Push query** : reçoit les changements en temps réel :

```
SELECT * FROM season_length_changes WHERE created_at < '2023-04-14 1
2:00:00' EMIT CHANGES;
```

- **Pull query** : récupère l'état actuel d'une table ou d'un titre précis :

```
SELECT * FROM season_length_change_counts WHERE title='Stranger Things';
```

4️⃣ Concepts clés à retenir

1. **Kafka Topics** : files de messages qui contiennent les événements.
2. **ksqlDB Stream** : flux de données continu, dérivé d'un topic.
3. **ksqlDB Table** : représentation de l'état actuel ou agrégé des données (matérialisé).
4. **Type personnalisé** : permet de structurer des colonnes complexes (ex : `before` et `after`).
5. **Push vs Pull** :
   - Push = flux continu, mise à jour en temps réel.
   - Pull = requête ponctuelle sur l'état actuel.
6. **Windowing (fenêtres temporelles)** : permet d'agréger les données sur des périodes définies (ex : 1h).