TP 3 : Streaming de ventes avec Kafka + Spark Structured Streaming + Delta Lake

Objectifs pédagogiques

- Simuler un flux de ventes en temps réel : Consommer avec Spark → Stocker dans Delta Lake → Analyser.
- Les résultats attendus :
  - Architecture streaming moderne (Kafka + Spark + Delta Lake)
  - Traitement temps réel avec Spark Structured Streaming
  - Stockage transactionnel avec Delta Lake (ACID, Time Travel)
  - Modèle Lakehouse (Bronze → Silver)
  - Production de données simulées réalistes
  - Supervision et monitoring de flux

PARTIE 1 — Prérequis

➢ Installer localement :

1. Apache Kafka (avec ZooKeeper)

- https://kafka.apache.org/quickstart
- Ou via brew install kafka

2. Spark 3.5+ avec support Delta Lake

- https://spark.apache.org/downloads.html
- Configurer SPARK_HOME, PYSPARK_PYTHON

3. Delta Lake

- pip install delta-spark

4. Python packages

- pip install kafka-python pyspark

PARTIE 2 — Producteur Kafka (simulateur de ventes en temps réel)

➢ **Énoncé :**

1. Fichier : producer_ventes.py

2. Lancer le producteur

- python producer_ventes.py
- Envoie une vente toutes les 2 secondes sur le topic ventes_stream.

PARTIE 3 — Consommateur Spark Streaming + Delta Lake

➢ **Énoncé :**

1. Fichier : spark_streaming_delta.py

2. Lancer le consommateur Spark

- spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.0,io.delta:delta-spark_2.12:3.2.0 spark_streaming_delta.py
- Adapte la version de Spark et Delta selon ton installation.

PARTIE 4 — Architecture Lakehouse (Bronze → Silver)

➢ Énoncé :

1. Fichier : spark_streaming_delta.py (batch ou micro-batch)

PARTIE 5 — Dashboard temps réel (optionnel)

➢ Énoncé :

1. Tu peux connecter **Power BI** ou **Grafana** à Delta Lake via :

- Databricks SQL Endpoint (si tu es sur Databricks)
- Trino/Presto avec connecteur Delta Lake
- Spark Thrift Server (pour SQL)
- Ou simplement lire la table Silver périodiquement :

```
# dashboard_query.py

df = spark.read.format("delta").load("/tmp/delta/silver/ventes_aggreges")

df.createOrReplaceTempView("ventes_silver")

top_clients = spark.sql("""

   SELECT pays, segment, SUM(total_depense) as ca_total

   FROM ventes_silver

   GROUP BY pays, segment

   ORDER BY ca_total DESC

""")

top_clients.show()
```

PARTIE 6 — Supervision & Monitoring

➢ Énoncé :

1. Ajoute dans ton script Spark :

```
   # Mesurer le débit

   df_enriched.withWatermark("timestamp", "10 minutes")...

   # Gérer les retards

   .option("withEventTime", "true")

   # Log des métriques

   query.recentProgres
```

Comment exécuter le TP sur PC ?

1. Démarrer ZooKeeper + Kafka

```
   # Terminal 1

   zookeeper-server-start /usr/local/etc/kafka/zookeeper.properties

   # Terminal 2

   kafka-server-start /usr/local/etc/kafka/server.properties

   # Terminal 3 — Créer le topic
```

kafka-topics --create --topic ventes_stream --bootstrap-server localhost:9092 --partitions 1 --replication-factor

2. Lancer le producteur

 # Terminal 4

python producer_ventes.py

3. Lancer le consommateur Spark

Terminal 5

spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.0,io.delta:delta-spark_2.12:3.2.0 spark_streaming_delta.py

4. (Optionnel) Lancer l'agrégation Silver

Terminal 6 (toutes les 5 min via cron, ou manuel)

python streaming_silver.py