PIPELINE BIG DATA POUR L'ANALYSE EN TEMPS RÉEL DE TRANSACTIONS E-COMMERCE

Encadré par : M. BADIR Hassan

Réalisé par : EL HARIRI kenza

Année universitaire: 2025/2026

Table de matières

INTRODUCTION

Dans le cadre du module Big Data / Data Engineering, ce projet a pour objectif la conception et la mise en œuvre d'un pipeline Big Data distribué complet, permettant le traitement de données à grande échelle en mode batch et en temps réel.
Avec l'augmentation continue du volume, de la variété et de la vélocité des données, les architectures centralisées deviennent insuffisantes. Les systèmes distribués basés sur des technologies telles que Apache Kafka, Apache Spark et les systèmes de stockage distribués constituent aujourd'hui une réponse adaptée à ces problématiques.
Ce projet s'inscrit dans cette logique en proposant une architecture intégrée couvrant l'ensemble du cycle de vie de la donnée :
ingestion → stockage → traitement → analyse → visualisation.

1.1 Contexte du projet

Le projet est réalisé dans un contexte pédagogique visant à rapprocher l'étudiant des pratiques professionnelles du Data Engineering.
Il met l'accent sur :

- La manipulation de flux de données temps réel.
- L'utilisation de frameworks distribués.
- La compréhension du rôle de chaque composant dans une architecture Big Data.

Le système développé repose sur une architecture modulaire, permettant la scalabilité, la tolérance aux pannes et la séparation claire des responsabilités entre les différentes briques technologiques.

1.2 Objectifs

Les principaux objectifs de ce projet sont :

- Concevoir une architecture Big Data distribuée cohérente
- Mettre en œuvre l'ingestion de données en temps réel via Kafka
- Traiter les données à l'aide de Apache Spark :
    - RDD
    - DataFrame
    - Spark SQL
    - Spark Streaming

- Implémenter du code Spark en Scala
- Stocker les données dans un système de stockage distribué
- Exécuter des requêtes SQL distribuées sur de grands volumes de données
- Proposer une visualisation exploitable des résultats
- Comprendre le fonctionnement global d’un pipeline Data Engineering industriel

ARCHITECTURE GÉNÉRALE DU SYSTÈME

Cette section présente l'architecture globale du système Big Data mis en place.
L'objectif est de fournir une vision d'ensemble claire et cohérente du pipeline, en identifiant le rôle de chaque composant et les interactions entre eux.
L'architecture adoptée suit un modèle distribué, modulaire et scalable, largement utilisé dans les environnements industriels de traitement de données massives.

2.1 Vue d'ensemble du pipeline Big Data

Le pipeline du système est organisé autour des étapes suivantes :

1.Ingestion des données
- Les données sont générées ou collectées en continu
- Elles sont envoyées vers un système de messagerie distribué

2.Traitement en temps réel
- Les flux de données sont consommés et traités en continu
- Des transformations, agrégations et filtrages sont appliqués

3.Stockage distribué
- Les données traitées sont stockées de manière persistante
- Le stockage est optimisé pour l'analyse à grande échelle

4.Analyse et requêtage
- Les données stockées sont interrogées via SQL distribué
- Des analyses analytiques sont effectuées

5.Visualisation
- Les résultats sont présentés sous forme de tableaux et graphiques
- L'objectif est de faciliter l'interprétation des données

Chaîne globale du pipeline :
Kafka → Spark Streaming → Stockage distribué → SQL distribué → Visualisation

2.2 Technologies et outils utilisés

L'architecture repose sur un ensemble de technologies complémentaires :

Apache Kafka

- Système de messagerie distribué
- Assure l'ingestion des données en temps réel
- Supporte un haut débit et la tolérance aux pannes

Apache Spark

- Moteur de traitement distribué
- Utilisation de :
    - RDD
    - DataFrame
    - Spark SQL
    - Spark Streaming
- Code écrit en Scala

Stockage distribué

- Utilisé pour la persistance des données
- Supporte le partitionnement et la compression
- Optimisé pour les traitements analytiques

SQL distribué

- Accès aux données via des requêtes SQL
- Support de tables externes et de requêtes massives

Outils de visualisation

- Analyse interactive
- Représentation graphique des résultats

DÉVELOPPEMENT ET MISE EN ŒUVRE

1.Infrastructure — Docker Compose

```yaml
services:
  # Zookeeper
  ▷Run Service
  zookeeper:
    image: wurstmeister/zookeeper:latest
    container_name: zookeeper
    ports:
      - "2181:2181"
    networks:
      - bigdata-network

  # Kafka
  ▷Run Service
  kafka:
    image: wurstmeister/kafka:latest
    container_name: kafka
    depends_on:
      - zookeeper
    ports:
      - "9092:9092"
      - "29092:29092"
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_LISTENERS: INTERNAL://0.0.0.0:9092,EXTERNAL://0.0.0.0:29092
      KAFKA_ADVERTISED_LISTENERS: INTERNAL://kafka:9092,EXTERNAL://localhost:29092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INTERNAL:PLAINTEXT,EXTERNAL:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: INTERNAL
      KAFKA_CREATE_TOPICS: "ecommerce-transactions:1:1"
    networks:
      - bigdata-network

  # Spark Master
  ▷Run Service
  spark-master:
    image: apache/spark:3.5.0-scala2.12-java11-python3-ubuntu
    container_name: spark-master
    ports:
      - "8080:8080"
      - "7077:7077"
      - "4040:4040"
    volumes:
      - ./spark-apps:/opt/spark-apps
      - ./data:/opt/data
    command: >
      bash -c "/opt/spark/bin/spark-class org.apache.spark.deploy.master.Master"
    networks:
      - bigdata-network

  # Spark Worker
  ▷Run Service
  spark-worker:
    image: apache/spark:3.5.0-scala2.12-java11-python3-ubuntu
    container_name: spark-worker
    depends_on:
      - spark-master
    environment:
```

DÉVELOPPEMENT ET MISE EN ŒUVRE

```yaml
# Spark Worker
▷Run Service
spark-worker:
  image: apache/spark:3.5.0-scala2.12-java11-python3-ubuntu
  container_name: spark-worker
  depends_on:
    - spark-master
  environment:
    SPARK_MASTER: spark://spark-master:7077
    SPARK_WORKER_CORES: 2
    SPARK_WORKER_MEMORY: 2G
  volumes:
    - ./spark-apps:/opt/spark-apps
    - ./data:/opt/data
  command: >
    bash -c "/opt/spark/bin/spark-class org.apache.spark.deploy.worker.Worker spark://spark-maste
  networks:
    - bigdata-network

# PostgreSQL
▷Run Service
postgres:
  image: postgres:15-alpine
  container_name: postgres
  ports:
    - "5432:5432"
  environment:
    POSTGRES_USER: bigdata
```

2.build.sbt

```
name := "ecommerce-streaming"

version := "1.0"

scalaVersion := "2.12.15"

libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-core" % "3.5.0",
  "org.apache.spark" %% "spark-sql" % "3.5.0",
  "org.apache.spark" %% "spark-streaming" % "3.5.0",
  "org.apache.spark" %% "spark-sql-kafka-0-10" % "3.5.0"
)
```

DÉVELOPPEMENT ET MISE EN ŒUVRE

```
PS C:\Users\hp\Desktop\bigdata-project> # Démarrer tous les containers
>> docker-compose up -d
>>
>> # Attendre que tout démarre (30 secondes)
>> Start-Sleep -Seconds 30
>>
>> # Vérifier que tout tourne
>> docker ps
>>
[+] Running 5/5
 ✓ Container zookeeper      Created
 ✓ Container postgres       Created
 ✓ Container spark-master   Created
 ✓ Container kafka          Created
 ✓ Container spark-worker   Created
```

```
CONTAINER ID   IMAGE                                          COMMAND                  CREATED
STATUS          PORTS                                                                   NAMES
2a19267b2655   apache/spark:3.5.0-scala2.12-java11-python3-ubuntu   "/opt/entrypoint.sh …"   3 days ago
Up 31 seconds                                                                          spark-worker
cfcf87e93939   apache/spark:3.5.0-scala2.12-java11-python3-ubuntu   "/opt/entrypoint.sh …"   3 days ago
Up 32 seconds   0.0.0.0:4040->4040/tcp, 0.0.0.0:7077->7077/tcp, 0.0.0.0:8080->8080/tcp  spark-master
d41bd643379f   wurstmeister/zookeeper:latest                  "/bin/sh -c '/usr/sb…"   3 days ago
Up 32 seconds   22/tcp, 2888/tcp, 3888/tcp, 0.0.0.0:2181->2181/tcp                      zookeeper
f54e8267537c   postgres:15-alpine                             "docker-entrypoint.s…"   3 days ago
Up 32 seconds   0.0.0.0:5432->5432/tcp                                                 postgres
PS C:\Users\hp\Desktop\bigdata-project>
```

DÉVELOPPEMENT ET MISE EN ŒUVRE

4. Spark Streaming — Exécution et Monitoring

Figure 5

Figure 6

DÉVELOPPEMENT ET MISE EN ŒUVRE

5. HDFS — Stockage et Organisation

Figure 7

6. Impala/Hive — Requêtes SQL Distribuées

Figure 8

DÉVELOPPEMENT ET MISE EN ŒUVRE

Figure 9

Figure 10

7. Jupyter — Analyse Interactive et Visualisation

Figure 11

Figure 12

Figure 13

Figure 14

RÉSULTATS ET ANALYSE DES PERFORMANCES