Devoir Big Data : Mise en place d'un Pipeline Big Data End-to-End pour l'Analyse du Trafic Urbain et de la Mobilité Intelligente

Introduction

Dans le cadre des **Smart Cities**, les villes modernes déploient des **capteurs urbains** (caméras, boucles magnétiques, capteurs IoT, applications mobiles) afin de collecter en continu des données liées au **trafic routier** et à la **mobilité des citoyens**.

Ces données permettent de :

- surveiller l'état du trafic en temps réel,
- détecter les congestions,
- analyser les flux de véhicules,
- améliorer la planification urbaine et la mobilité.

La problématique centrale de ce projet est la suivante :

Comment concevoir et implémenter un pipeline Big Data capable de collecter des données de trafic urbain en temps réel, de les stocker dans un Data Lake, de les traiter efficacement, puis de produire des indicateurs exploitables pour la gestion intelligente de la mobilité, tout en assurant l'automatisation et la supervision du pipeline ?

Dans ce devoir, vous jouez le rôle d'un **Data Engineer** travaillant sur une plateforme **Smart City**.

Besoin métier :

La municipalité souhaite disposer d'un système permettant de :

- suivre le **niveau de trafic en temps réel**
- identifier les **zones congestionnées**
- analyser le trafic par :
    - zone
    - période
    - type de voie
- exploiter les données pour la **prise de décision urbaine**

Nature des données à traiter

Les données sont des **événements de trafic routier**, générés par des capteurs urbains.

Chaque événement représente **une mesure instantanée du trafic** sur un axe routier.

Structure obligatoire d'un événement de trafic (format JSON)

Chaque événement doit contenir **au minimum** :

- **sensor_id** : identifiant du capteur
- **road_id** : identifiant de la route
- **road_type** : type de route (autoroute, avenue, rue)
- **zone** : zone géographique (quartier, secteur)
- **vehicle_count** : nombre de véhicules détectés
- **average_speed** : vitesse moyenne (km/h)
- **occupancy_rate** : taux d'occupation (%)
- **event_time** : date et heure de la mesure

Génération des données

Les données **ne sont pas fournies**.
Vous devez **simuler un réseau de capteurs de trafic**.

Ce que vous devez faire :

- Créer un **script de génération de données** (Python ou Java)
- Générer des événements :
    - de manière continue
    - avec des valeurs réalistes
- Simuler :
    - plusieurs capteurs
    - plusieurs routes
    - plusieurs zones
- Générer **des milliers d'événements**

➢ *Objectif : simuler un système Smart City réel.*

Étape 1 — Collecte des données (Data Collection)

Ce que vous devez faire :

- Implémenter le **générateur de données de trafic**
- Produire les événements au **format JSON**
- Vérifier la cohérence des valeurs :

    - vitesse réaliste
    - trafic variable selon l'heure
- Documenter la structure des événements

➢ Objectif : simuler des capteurs urbains réels.

Étape 2 — Ingestion des données (Data Ingestion)

Les données doivent être ingérées **en temps réel**.

Ce que vous devez faire :

- Mettre en place une ingestion streaming avec **Apache Kafka**
- Créer un topic Kafka (ex. traffic-events)
- Développer un producer Kafka
- Tester l'ingestion avec un consumer
- Expliquer :
    - le partitionnement
    - la fréquence d'envoi
    - le volume de données

➢ Objectif : gérer un flux temps réel IoT.

Étape 3 — Stockage des données brutes (Data Lake – Raw Zone)

Ce que vous devez faire :

- Utiliser **HDFS** comme Data Lake
- Créer :
    - /data/raw/traffic
- Consommer les messages Kafka et les écrire dans HDFS
- Organiser les données par date et zone
- Vérifier le stockage

➢ Objectif : construire une zone *raw* Smart City.

Étape 4 — Traitement des données (Data Processing)

Ce que vous devez faire :

- Utiliser **Apache Spark**
- Lire les données depuis /data/raw/traffic

- Calculer :
    - trafic moyen par zone
    - vitesse moyenne par route
    - taux de congestion
- Identifier les **zones à forte congestion**
- Sauvegarder les résultats dans :
    - /data/processed/traffic

➢ Objectif : analyser des données de mobilité.

Étape 5 — Structuration analytique (Analytics Zone)

Ce que vous devez faire :

- Sauvegarder les résultats au format **Parquet**
- Créer :
    - /data/analytics/traffic
- Justifier le format analytique
- Vérifier les fichiers

➢ Objectif : préparer les données pour l'analyse avancée.

Étape 6 — Exploitation et visualisation

Ce que vous devez faire :

- Définir des **KPI de mobilité** :
    - trafic par zone
    - vitesse moyenne
    - taux de congestion
- Créer des dashboards dans **Grafana**
- Visualiser :
    - évolution du trafic
    - zones critiques
- Interpréter les résultats

➢ Objectif : aider à la décision urbaine.

Étape 7 — Orchestration du pipeline

Ce que vous devez faire :

- Mettre en place **Apache Airflow**
- Créer un DAG :
    - ingestion
    - traitement Spark
    - validation
- Gérer les dépendances
- Lancer le pipeline

➢ Objectif : automatiser un pipeline Smart City.