Documentation de l’architecture

Dans ce projet, nous avons pour objectif de traiter et intégrer des données provenant de plusieurs sources (Kafka et HDFS) en temps réel, tout en calculant des métriques utiles pour analyser l'impact du COVID-19 sur les hôpitaux.

1. **Kafka Producer (flux de données en streaming)** :
    - Le fichier `hospital-utilization-trends` est injecté en streaming dans Kafka. Nous avons créé un **Kafka Producer** pour lire le fichier par séries de 100 lignes et les envoyer à Kafka toutes les 10 secondes.
    - Ce processus permet de simuler un flux de données provenant d’un client externe.

2. **Kafka Consumer (réception des données)** :
    - Le **Kafka Consumer** écoute les messages envoyés par le Kafka Producer et récupère les données en temps réel.
    - Chaque message reçu est ensuite intégré dans le système de stockage pour être analysé et traité.

3. **HDFS (stockage des données historiques)** :
    - Les deux autres fichiers (`in-hospital-mortality-trends-by-diagnosis-type` et `in-hospital-mortality-trends-by-health-category`) sont lus depuis **HDFS** (Hadoop Distributed File System), où les données sont stockées de manière distribuée et accessible pour les traitements Spark.

4. **Spark Streaming pour le traitement des données** :
    - Nous utilisons **Spark Streaming** pour consommer les données depuis Kafka et effectuer des transformations sur ces données en temps réel.

- Spark permet de gérer les flux de données en continue, de les intégrer avec celles provenant de HDFS et de calculer les métriques nécessaires.

5. **Base de données pour le stockage final** :

   - Nous avons choisi **PostgreSQL** pour stocker les résultats finaux. Cette base de données est idéale pour le stockage de **données temporelles** et permet des requêtes efficaces. PostgreSQL offre également une grande stabilité et permet de faire des analyses complexes sur les données agrégées.

6. **Gestion des versions et intégrité des données** :

   - Le système doit permettre de revenir à une version antérieure des données en cas de besoin (par exemple, en cas de problème de traitement). Nous gérons cela avec un **contrôle de version** au niveau des tables dans PostgreSQL.
   - Si des incohérences sont détectées dans les données consommées, nous avons prévu un mécanisme de **retraitement** des données avec Spark, en relisant les messages Kafka et en réintégrant les données traitées.