Data Pipeline Architecture

1. Objectifs du pipeline de données

- **But principal** : Assurer un flux de données cohérent, sécurisé, et en temps réel entre les systèmes OLTP, OLAP, et NoSQL pour supporter la gestion des transactions, l'analyse avancée, et les cas d'utilisation basés sur des données non structurées.

- **Exigences principales** :

    - **Ingestion de Données Temps Réel & Batch** : Gérer l'ingestion de données transactionnelles et analytiques à partir de multiples sources.
    - **Transformation et Enrichissement** : Nettoyer, normaliser et enrichir les données avant qu'elles soient intégrées dans les différents systèmes de stockage.
    - **Sécurité et Conformité** : Appliquer des contrôles de sécurité pour le chiffrement, l'accès aux données, et assurer la conformité aux réglementations (GDPR, PCI-DSS).

2. Architecture globale du pipeline de données

- **Ingestion de Données**

    - Utiliser **Apache Kafka** comme **outil de streaming** pour l'ingestion de données en temps réel provenant de diverses sources (OLTP, logs de services, API externes).
    - Configurer des topics Kafka pour catégoriser les types de données.
    - **Connecteurs Kafka** pour intégrer les flux de données en temps réel vers les systèmes de stockage (NoSQL, Data Lake).

- **Stockage des Données Brutes**

    - **Data Lake (Amazon S3)** : Stocker les données brutes et semi-structurées (e.fichiers JSON, logs) pour une ingestion facile dans les systèmes d'analytique et de machine learning.
    - Intégrer **Apache Hudi** ou **Delta Lake** pour garantir la **gestion des versions**, **la qualité des données**, et permettre les **mises à jour et suppressions** en temps réel (important pour la conformité aux réglementations comme GDPR).

- **Transformation des Données**

    - Utiliser **Apache Spark** pour la transformation de données à grande échelle. Spark permet la **manipulation des données batch et en temps réel** (via Spark Streaming).

- **Transformation ETL** (Extract, Transform, Load) :
  - Extraire les données brutes du Data Lake ou des systèmes OLTP.
  - Nettoyer et normaliser les données (suppression des doublons, transformation des formats de dates, etc.).
  - Enrichir les données avec des informations supplémentaires (géolocalisation basée sur l'adresse IP).

- **Stockage des données Transformées**

  - **OLAP System (Snowflake, BigQuery)** : Une fois transformées, les données sont chargées dans un système OLAP pour une analyse rapide et des requêtes complexes.
  - Mettre en place des **tables agrégées** ou des **vues matérialisées** pour accélérer les requêtes fréquentes et les rapports complexes.

- **Stockage des données NoSQL**

  - **NoSQL System (MongoDB)** : Les données non structurées ou semi-structurées (logs d'activité, caractéristiques pour le ML) sont stockées dans une base NoSQL.
  - Les documents NoSQL sont indexés par des champs clés comme `Transaction_ID`, `User_ID`, ou `Timestamp` pour faciliter des recherches rapides.

3. Technologies clés et outils

- **Apache Kafka** :

  - **Rôle** : Streaming de données temps réel, ingestion de données provenant de multiples sources, et distribution vers les cibles (OLAP, NoSQL, Data Lake).
  - **Avantages** : Haute disponibilité, scalabilité, et capacité à gérer des volumes importants de données en temps réel.

- **Apache Spark** :

  - **Rôle** : Transformation et enrichissement des données pour les pipelines ETL/ELT. Supporte des tâches batch et streaming.
  - **Avantages** : Haute performance sur les transformations de données volumineuses et compatibilité avec une variété de sources de données (S3, JDBC, NoSQL).

- **Data Lake (Amazon S3)** :

  - **Rôle** : Stockage centralisé des données brutes et semi-structurées pour la persistance et la préparation à des analyses futures.
  - **Avantages** : Scalabilité, faible coût de stockage, intégration facile avec de nombreux outils analytiques et de traitement de données.

- **OLAP system (Snowflake, Google BigQuery)** :
    - **Rôle** : Support de l'analyse de données, des requêtes complexes, et des rapports ad hoc sur de grandes quantités de données agrégées.
    - **Avantages** : Optimisation des requêtes pour des analyses rapides, support des joints et des agrégations complexes.

- **NoSQL system (MongoDB)** :
    - **Rôle** : Stockage et gestion des données non structurées, semi-structurées et des caractéristiques utilisées pour les modèles ML.
    - **Avantages** : Flexibilité de schéma, requêtes rapides sur des champs indexés, intégration avec d'autres outils analytiques et de machine learning.

4. Sécurité et Conformité dans le pipeline de données

- **Chiffrement des données** :
    - **Chiffrement des Données en Transit** : Toutes les données transférées dans le pipeline (entre Kafka, Data Lake, OLAP, NoSQL) sont chiffrées avec TLS.
    - **Chiffrement des Données au Repos** : Les données stockées dans le Data Lake, OLAP, et NoSQL sont chiffrées avec AES-256.

- **Contrôles d'accès** :
    - **Authentification et autorisation** : Mettre en place un contrôle basé sur les rôles (RBAC) pour restreindre l'accès aux données selon le rôle des utilisateurs.
    - **MFA (Multi-Factor Authentication)** : Activer l'authentification multi-facteurs pour les accès aux données sensibles.

- **Audit et journaux d'activité** :
    - **Audit des accès aux données** : Enregistrer tous les accès et modifications des données dans des logs sécurisés pour assurer une traçabilité complète.

5. Synchronisation et consistance des données

- **Capture de Changements de Données (CDC)** :
    - Mettre en place un mécanisme de **CDC** pour assurer que toutes les modifications dans les systèmes OLTP (comme les transactions de paiement) sont synchronisées avec les systèmes OLAP et NoSQL.
    - Utiliser des outils comme **Debezium** ou **AWS Database Migration Service (DMS)** pour capturer et diffuser les changements.

- **Gestion des mises à jour temps réel** :
    - Les pipelines de données doivent **supporter les mises à jour en temps réel** pour garantir que les systèmes OLAP et NoSQL disposent des informations les plus à jour pour les analyses et les modèles de prédiction.

6. Surveillance et surveillance des performances

- **Surveillance des pipelines de données** :
    - Utiliser des outils de monitoring comme **Prometheus** pour surveiller la latence, le débit, et la santé des pipelines de données.
    - **Configurer des alertes** pour détecter les anomalies dans les flux de données ou les erreurs de traitement.

- **Suivi des performances des modèles ML** :
    - Pour les modèles de machine learning déployés, **surveiller la latence des prédictions et les éventuelles dégradations de performance** (drift des données).

7. Automatisation des pipelines et orchestration

- **Orchestration des pipelines ETL/ELT** :
    - Utiliser un outil comme **Apache Airflow** ou **Prefect** pour orchestrer les tâches ETL, gérer les dépendances, planifier les transformations, et assurer une exécution correcte des pipelines.
    - **Gestion des Dépendances** : Définir les flux de travail et les dépendances entre les tâches pour garantir que les transformations et chargements de données se produisent dans le bon ordre.

8. Plan de redondance et de reprise sur incident

- **Récupération après incident** :
    - Configurer des **mécanismes de redondance et de reprise après incident** (backup et restauration) pour garantir la disponibilité continue des données et la résilience du pipeline de données.
    - **Snapshot et Sauvegarde** : Mettre en place des sauvegardes régulières des systèmes OLAP et NoSQL, ainsi que des snapshots pour les données critiques dans le Data Lake.