Rapport détaillé : Pipeline ETL avec Kafka et Spark

# 1. Introduction

L'objectif de ce projet était de concevoir un pipeline de traitement de données en temps réel et par lots en utilisant **Kafka** pour la gestion des flux de données et **Spark** pour le traitement de ces données. Le pipeline devait permettre la collecte de données sur des étudiants (nom, prénom, âge, filière), leur transformation, leur analyse (statistiques), et leur sauvegarde dans un fichier CSV. Ce projet couvre un scénario :

1. **Traitement en temps réel** avec Spark Streaming pour la consommation continue de données.

# 2.Environnement et Technologies Utilisées

1. **Kafka sur Aiven :**

* Aiven est utilisé pour gérer la plateforme Kafka de manière managée, offrant une solution simplifiée pour la gestion des clusters Kafka. Cela permet une administration plus facile, sans souci de maintenance des infrastructures sous-jacentes.
* Kafka est utilisé pour l'envoi et la consommation des données en temps réel à travers un topic dédié, nommé etudiants\_topic. Les producteurs envoient des messages dans ce topic, et les consommateurs traitent ces messages en temps réel. Aiven garantit une haute disponibilité et une scalabilité optimale de Kafka, assurant un traitement fiable des données.

1. **Apache Spark :**

* PySpark, la version Python de Apache Spark, est utilisée pour le traitement des données en temps réel via Spark Streaming.
* L'intégration avec Kafka se fait dans un environnement de développement local, en utilisant VSCode pour le développement du code Python. Spark consomme des données en temps réel provenant de Kafka, applique des transformations et réalise des analyses statistiques sur les données traitées. Cette approche permet d'effectuer un traitement en continu des flux de données en temps réel, en exploitant les puissantes capacités de calcul distribuées de Spark.

1. **Technologies de Sécurité :**

* Les communications entre le producteur et le consommateur Kafka sont sécurisés grâce à l'utilisation de SSL. Les certificats nécessaires pour garantir la sécurité des échanges de données sont fournis sous forme de fichiers ca.pem, service.cert, et service.key. Cela permet d’assurer que les échanges entre les différents composants du pipeline sont sécurisés et que les données sensibles sont protégées.

1. **Python :**

* Le langage Python est utilisé pour développer l'ensemble du pipeline ETL, y compris la génération de données (producteur Kafka), la consommation et le traitement des données (consommateur Kafka avec Spark), et la sauvegarde des résultats. Les bibliothèques comme Faker (pour générer des données aléatoires) et csv (pour la gestion des fichiers CSV) sont également utilisées pour faciliter le développement.

# 3.Étapes du Pipeline ETL avec Spark

Le pipeline ETL (Extract, Transform, Load) est un processus clé dans le traitement des données, qui permet d'extraire, de transformer et de charger les données d'un format à un autre. Dans le cadre de ce projet, un pipeline ETL a été conçu pour traiter un fichier de données sur les étudiants à l'aide de Apache Spark. Ce pipeline comporte quatre étapes principales : extraction des données, transformation des données, calcul des statistiques, et chargement des données transformées. Voici un aperçu de chaqueétape :

**1. Extraction des données**

Le processus commence par le chargement des données à partir d'un fichier CSV contenant les informations des étudiants, telles que le nom, le prénom, l'âge et la filière. Spark facilite ce processus grâce à la fonction **spark.read.csv()**, qui permet de charger les données dans un **DataFrame Spark** tout en garantissant que le format des colonnes soit automatiquement détecté grâce à **inferSchema**.

A screenshot of a computer

Description automatically generated

**2. Transformation des données**

Une fois les données extraites, la transformation consiste en deux opérations clés :

Normalisation des noms : Les noms des étudiants sont convertis en majuscules pour uniformiser le format des données, facilitant ainsi toute analyse ou comparaison ultérieure. Cela se fait avec la fonction **withColumn(),** qui permet de modifier les colonnes du DataFrame.

Filtrage des étudiants : Ensuite, un filtrage est effectué pour ne conserver que les étudiants âgés de plus de 20 ans, à l'aide de la fonction **filter().**



A white rectangular object with a black border

Description automatically generated with medium confidence

**3. Calcul des statistiques**

Une étape importante du pipeline consiste à calculer la moyenne d'âge par filière. Grâce à la fonction **groupBy()** suivie de l'agrégation **avg(),** nous pouvons regrouper les étudiants selon leur filière et calculer l'âge moyen pour chaque groupe. Cette statistique est cruciale pour obtenir des informations sur les profils démographiques des différentes filières d'études.

A computer screen shot of a computer code

Description automatically generated

**4. Chargement des données transformées**

Enfin, une fois les données transformées et les statistiques calculées, celles-ci sont sauvegardées dans un nouveau **fichier CSV**. La méthode **write.csv()** permet d'écrire ces données dans un format structuré, prêt à être utilisé pour des analyses futures ou à être intégré dans d'autres systèmes.

A computer screen shot of a computer code

Description automatically generated

# 4. Intégration avec Kafka et Spark Streaming

**1. Producteur Kafka (Producer.py) :**

Le **producteur Kafka** génère des données simulées d’étudiants. Il utilise la bibliothèque **Faker** pour créer des noms, prénoms, âges et filières aléatoires. Ces données sont envoyées dans un topic Kafka nommé etudiants\_topic sur Aiven.

**Résultats de terminal :**

**A screen shot of a computer

Description automatically generatedRésultats sur Aiven:  
A screenshot of a computer

Description automatically generated**

**2. Consommateur Spark Streaming**

Le consommateur a été implémenté pour lire les données en continu depuis Kafka. Il transforme les données reçues en DataFrame Spark et applique des transformations :

* Conversion des noms en majuscules.
* Filtrage des étudiants de plus de 20 ans.
* Calcul de la moyenne d'âge par filière.

**A screenshot of a computer

Description automatically generatedRésultats de terminal :**

**4. Défis rencontrés et solutions proposées**

**Défis techniques :**

1. **Erreur liée à kafka.vendor.six.moves :**
   * **Problème :** L'absence de la dépendance **six** causait une erreur à l’exécution.
   * **Solution :** Migration vers la bibliothèque **confluent\_kafka** plus robuste et mieux adaptée.
2. **Configuration SSL :**
   * **Problème :** Problèmes de connexion à Kafka dus à des erreurs dans les chemins des certificats SSL.
   * **Solution :** Vérification et correction des fichiers de configuration, et utilisation de fichiers certifiés par Aiven.

**Autres difficultés :**

* Synchronisation entre le producteur Kafka et le consommateur Spark Streaming.
* Gestion des interruptions du consommateur.