Au cours de cet examen qui consistais à la Création d'un pipeline de données avec Apache Spark et Kafka  
  
Le travail était assez simple mais comme toujours rien n’est acquis dans un travail.

Pour débuter j’ai eu créer un compte Aiven, car pour commencer ma partie code j’en avais besoin afin de mener à bien mon travail

Sur Aiven j’ai pris les informations principales qu’il m’a été donné

Service URL de mon compte

J’ai du créer un topics sur Aiven,   
j’ai recu des Acces key, Acces Certificate, CA Certificate dans la rubrique Overviews

Pour mener à bien et pour réaliser la partie pratique j’ai utilisé Google Collab

J’ai commencé par la configuration de PysSpark

Installation de PySpark

Configuration de Spark

Simulation de flux de données

Création d’une dataframe nommé students.csv

Lecture du fichier csv

Convertions des noms en majuscule

Filtrage des étudiants de plus de 20 ans

Calcul de la moyenne d’age

Ensuite nous avons viré à Installation de Kafka

Configuration du producer

Configuration de Spark Streaming

Installation de confluent-kafka et de Kafka-python

Configuration du consumer

Transformation des noms en majuscules

Filtrage des étudiants de plus de 20 ans

Calcul statistiques moyenne d’age par filiere

Une petite visualisation de notre bases de données

Boxplot

Bar Plot

Pie Chart

**Rapport d'examen : Création d'un pipeline de données avec Apache Spark et Kafka**

**Introduction**

Dans le cadre de cet examen, l'objectif principal était de concevoir et de mettre en œuvre un pipeline de données en intégrant le traitement par lots (batch) et le traitement en flux (streaming) à l'aide d'Apache Spark et Kafka. Ce projet a permis de simuler un cas d'usage pratique d’ingénierie des données où des informations sur des étudiants sont extraites, transformées, analysées et sauvegardées en temps réel.

Ce rapport présente en détail les étapes réalisées, les configurations nécessaires, les transformations appliquées aux données, les statistiques générées, ainsi que les défis rencontrés et les solutions proposées.

**1. Configuration de l’environnement**

La première étape a consisté à configurer l’environnement technique pour permettre l’exécution des différents outils requis :

* **Création d’un compte Aiven** : Ce service cloud a été utilisé pour obtenir un cluster Kafka managé. Les éléments suivants ont été configurés :
  + **Service URL** : L’adresse de connexion au cluster Kafka.
  + **Certificats** : Accès Key, Accès Certificate, et CA Certificate, nécessaires pour sécuriser les communications Kafka.
  + **Création d’un topic Kafka** : Un topic nommé student-data a été créé pour publier et consommer les données.
* **Google Colab** : Cet environnement a été utilisé pour exécuter tout le code, en raison de sa simplicité d’utilisation et de son support des bibliothèques nécessaires.

Les outils suivants ont été installés et configurés :

* **PySpark** : Installation et configuration de Spark pour les transformations et analyses des données.
* **Kafka-python** et **Confluent-kafka** : Pour interagir avec Kafka et gérer les flux de données en temps réel.

**2. Pipeline ETL : Traitement par lots**

Un fichier CSV contenant des informations sur 20 étudiants a été utilisé pour simuler une source statique. Les étapes suivantes ont été réalisées :

1. **Lecture des données** : Les données du fichier students.csv ont été chargées dans un DataFrame Spark.
2. **Transformations appliquées** :
   * Conversion des noms des étudiants en majuscules pour uniformiser les données.
   * Filtrage des étudiants ayant plus de 20 ans pour répondre à une exigence spécifique.
3. **Analyse statistique** :
   * Calcul de la moyenne d'âge des étudiants par filière pour extraire des insights pertinents.

Les données transformées ont été sauvegardées dans un nouveau fichier CSV et visualisées dans Colab pour vérification.

**3. Pipeline de streaming : Intégration avec Kafka et Spark Streaming**

Le pipeline en flux de données a permis de traiter des données en temps réel simulant l’ajout progressif d'étudiants.

1. **Producteur Kafka** :  
   Un script Python a été créé pour envoyer des messages contenant les informations des étudiants au topic student-data. Les données étaient envoyées sous forme de JSON avec un intervalle simulé pour imiter un flux en temps réel.
2. **Consommateur Kafka avec Spark Streaming** :  
   Un consommateur Spark Streaming a été implémenté pour :
   * Lire les messages Kafka en temps réel.
   * Convertir les données reçues en DataFrame Spark pour les traiter immédiatement.
   * Appliquer les mêmes transformations que dans le traitement par lots :
     + Conversion des noms en majuscules.
     + Filtrage des étudiants ayant plus de 20 ans.
     + Calcul de la moyenne d’âge par filière.
   * Sauvegarder les résultats dans un fichier CSV et les visualiser.

**4. Visualisation et analyse des résultats**

Une fois les données transformées et sauvegardées, plusieurs visualisations ont été réalisées pour mieux comprendre les résultats :

* **Boxplot** : Pour analyser la distribution des âges par filière.
* **Bar plot** : Pour afficher la répartition des étudiants par filière.
* **Pie chart** : Pour illustrer la part relative de chaque filière.

Ces visualisations ont permis de valider la cohérence des données traitées et d'extraire des insights utiles.

**5. Défis rencontrés et solutions proposées**

* **Configuration de Kafka sur Aiven** :
  + Défi : La gestion des certificats et la connexion sécurisée au cluster Kafka ont nécessité une attention particulière.
  + Solution : Une documentation détaillée a été suivie pour configurer les certificats correctement.
* **Gestion des dépendances dans Colab** :
  + Défi : Certaines dépendances, comme Kafka-python et Confluent-kafka, ont nécessité des configurations spécifiques.
  + Solution : Toutes les bibliothèques ont été installées via pip en suivant un ordre précis pour éviter les conflits.
* **Streaming et latence** :
  + Défi : Garantir un traitement fluide des données en temps réel tout en minimisant la latence.
  + Solution : Optimisation des paramètres de Spark Streaming et Kafka pour une meilleure performance.

**6. Exécution du pipeline complet**

Une démonstration fonctionnelle a été réalisée pour montrer :

1. **Traitement par lots** : Lecture et transformation des données statiques (CSV).
2. **Traitement en flux** :
   * Production des données en temps réel vers Kafka.
   * Consommation et transformation en direct avec Spark Streaming.
3. **Visualisation et sauvegarde** : Les résultats ont été sauvegardés et visualisés pour validation.

**Conclusion**

Ce projet a permis de mettre en œuvre un pipeline de données complet, intégrant les concepts clés de l’ingénierie des données, notamment l’ETL et le traitement en flux. La maîtrise des outils Apache Spark et Kafka a été renforcée, et le projet a démontré l'importance des pipelines pour traiter efficacement des données en temps réel.

Ce travail peut servir de base pour des projets plus complexes, comme l’analyse prédictive ou la gestion de grandes bases de données en temps réel.