Rapport : Création d'un Pipeline de Données avec Apache Spark et Kafka

Partie 1 : Configuration et Préparation

Installation et Configuration :

1. **Environnement installé :**
    - PySpark a été installé avec la commande !pip install pyspark.
    - Kafka-python et Confluent Kafka ont été installés pour la gestion des messages Kafka (!pip install kafka-python et !pip install confluent-kafka).
2. **Configuration de Spark :**
    - Une session Spark a été initialisée pour effectuer les transformations et analyses des données :
    - spark = SparkSession.builder.appName("ETL_Students").getOrCreate()
3. **Configuration de Kafka :**
    - Un serveur Kafka distant a été configuré avec SSL pour garantir une communication sécurisée entre le producteur et le consommateur.
    - Les fichiers de certificat nécessaires (ca.pem, service.cert, service.key) ont été utilisés.
4. **Fichiers CSV :**
    - Un fichier CSV etudiants.csv a été généré avec des données simulées représentant les informations des étudiants (nom, prénom, âge, filière). Ce fichier a été créé avec Python et vérifié avec Pandas pour s'assurer de sa structure correcte.

Partie 2 : Pipeline ETL avec Spark

Extraction des données :

Les données ont été extraites du fichier CSV etudiants.csv à l'aide de PySpark :

df = spark.read.csv('etudiants.csv', header=True, inferSchema=True)

Transformation des données :

1. **Conversion en majuscules :**
    - Les colonnes Nom et Prenom ont été transformées pour apparaître en majuscules :
    - df_transformed = df.withColumn("Nom", upper(col("Nom")))
2. **Filtrage :**
    - Les étudiants de plus de 20 ans ont été sélectionnés :
    - df_filtered = df_transformed.filter(col("Age") > 20)

3. **Calcul des statistiques :**
    - La moyenne d'âge a été calculée par filière à l'aide de la fonction d'agrégation Spark :
    - df_stats = df_filtered.groupBy("Filiere").agg(avg("Age").alias("Moyenne_Age"))

Chargement des données :

Les résultats transformés et les statistiques ont été enregistrés dans un fichier CSV nommé resultats_etudiants.csv :

df_stats.write.csv('resultats_etudiants.csv', header=True)

Partie 3 : Intégration avec Kafka et Spark Streaming

Création d'un producteur Kafka :

1. Un producteur Kafka a été configuré pour envoyer des données simulées au topic etudiants :
2. producer = KafkaProducer(
3. bootstrap_servers='kafka-18536ef0-isy03436-05ce.g.aivencloud.com:14686',
4. security_protocol='SSL',
5. ssl_cafile='/content/ca.pem',
6. ssl_certfile='/content/service.cert',
7. ssl_keyfile='/content/service.key',
8. value_serializer=lambda v: json.dumps(v).encode('utf-8')
9.)
10. Les messages envoyés incluent les noms, prénoms, âge et filière, par exemple :
11. Message envoyé : {'Nom': 'Jean', 'Prenom': 'Ben', 'Age': 30, 'Filiere': 'IA'}

Création d'un consommateur Kafka :

1. Le consommateur Kafka a été configuré pour recevoir les messages du topic etudiants.
2. Les messages consommés ont été transformés (noms en majuscules, filtrage des étudiants de plus de 20 ans) et enregistrés dans une liste Python pour analyse :
3. if age > 20:
4. transformed_data.append({
5. 'first_name': first_name.upper(),
6. 'last_name': last_name.upper(),
7. 'age': age,
8. 'field': field
9. })

Traitement et analyse des données :

1. La moyenne d'âge par filière a été calculée après transformation des données consommées.
2. Les résultats ont été sauvegardés dans un fichier CSV : etudiants_transformed.csv.

Partie 4 : Reporting et Analyse

Configuration de l'environnement :

- Spark et Kafka ont été correctement configurés pour exécuter le pipeline de données en temps réel.
- La communication entre le producteur et le consommateur Kafka a été testée avec succès.

Transformations appliquées :

- Transformation des noms en majuscules.
- Filtrage des étudiants ayant plus de 20 ans.
- Calcul des statistiques sur les données transformées (moyenne d'âge par filière).

Analyses et statistiques générées :

Exemple des résultats obtenus :

```
+------------+------------------+
|    Filiere|      Moyenne_Age|
+------------+------------------+
|Informatique|          24.875|
|    Gestion|            25.0|
|    Finance| 23.77777777777778|
|      Droit|26.181818181818183|
|         IA|           25.125|
+------------+------------------+
```

Défis rencontrés et solutions :

1. **Problème :** Gestion des connexions Kafka SSL.
    - **Solution :** Utilisation des fichiers de certificats appropriés pour la communication sécurisée.
2. **Problème :** Lenteur dans la boucle de consommation Kafka.
    - **Solution :** Réduction de l'attente de poll() et ajout d'une limite de durée d'exécution.

Conclusion

Un pipeline complet a été conçu et mis en œuvre avec succès, intégrant le traitement en lots (Spark) et le traitement en temps réel (Kafka). Les transformations, filtrages et analyses répondent aux objectifs du projet. Les données sont traitées et sauvegardées avec fiabilité pour permettre une prise de décision rapide et efficace.

Livrables

1. Code source du producteur et consommateur Kafka.
2. Script ETL pour PySpark.
3. Fichiers CSV générés :
    - resultats_etudiants.csv
    - etudiants_transformed.csv.
4. Rapport détaillé (ce document).