**Rapport sur la Création d'un Pipeline de Données avec Apache Spark et Kafka**

**Objectif de l'Atelier**

L'objectif de cet atelier est de concevoir, développer et implémenter un pipeline de données complet, en intégrant à la fois le traitement par lots (batch) et le traitement en flux (streaming) à l'aide des technologies Apache Spark et Apache Kafka. Le pipeline doit être capable d'extraire, de transformer et de charger (ETL) des données à partir de fichiers CSV et de flux de données en temps réel, tout en permettant une analyse des données en temps réel.

**Partie 1 : Configuration et Préparation**

**1.1 Installation de l'Environnement**

L'environnement nécessaire pour exécuter Apache Spark et Apache Kafka a été configuré sur une machine locale (Google Colab pour ce projet).

* **Java** : Java est installé et configuré sur la machine locale, nécessaire pour exécuter Apache Spark.
* **Apache Spark** : Spark a été installé et intégré avec PySpark pour le traitement des données.
* **Apache Kafka** : Kafka est installé et configuré pour émettre et consommer des données en temps réel. La communication avec Kafka se fait par un client Python Kafka.

**1.2 Démarrage des Services**

Les services nécessaires à l'exécution d'Apache Kafka (ZooKeeper et Kafka Broker) ont été démarrés. Après avoir démarré les services, il a été vérifié que Kafka fonctionne correctement en émettant et en consommant des données.

**Partie 2 : Pipeline ETL avec Apache Spark**

**2.1 Extraction des Données**

Les données sur les étudiants ont été extraites à partir d'un fichier CSV. Ce fichier contient les informations suivantes : nom, prénom, âge et filière des étudiants. Le fichier a été chargé dans un DataFrame Spark en utilisant PySpark.

python

Copier le code

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("ETL Students").getOrCreate()

df = spark.read.csv("/content/drive/MyDrive/Colab Notebooks/etudiants.csv", header=True, inferSchema=True)

df.show()

Le DataFrame initial contient les informations suivantes :

| **nom** | **prénom** | **âge** | **filière** |
| --- | --- | --- | --- |
| Benkirane | Sarah | 21 | Informatique |
| ElAmrani | Ahmed | 19 | Gestion |
| Hamdani | Salma | 22 | Informatique |
| Tazi | Omar | 20 | Marketing |
| ... | ... | ... | ... |

**2.2 Transformation des Données**

Des transformations ont été appliquées aux données extraites.

1. **Conversion des noms en majuscules** : Tous les noms des étudiants ont été convertis en majuscules pour normaliser les données.

python

Copier le code

from pyspark.sql.functions import upper, col

df1 = df.withColumn("nom", upper(col("nom")))

df1.show()

1. **Filtrage des étudiants de plus de 20 ans** : Seuls les étudiants ayant plus de 20 ans ont été retenus pour l'analyse.

python

Copier le code

df2 = df1.filter(df1["age"] > 20)

df2.show()

1. **Calcul des statistiques** : La moyenne d'âge par filière a été calculée pour obtenir des informations sur la distribution des âges parmi les différents départements.

python

Copier le code

from pyspark.sql.functions import avg

stats = df2.groupBy("filiere").agg(avg("age").alias("moyenne\_age"))

stats.show()

**2.3 Chargement des Données**

Les données transformées ont été sauvegardées dans un fichier CSV pour un stockage ultérieur.

python

Copier le code

df2.write.csv("/content/etudiants\_transforme.csv", header=True)

**Partie 3 : Intégration avec Kafka et Spark Streaming**

**3.1 Création d'un Producteur Kafka**

Un producteur Kafka a été créé pour envoyer des données en temps réel dans un topic Kafka. Chaque ligne du fichier CSV contenant les informations des étudiants est envoyée comme un message JSON.

python

Copier le code

from kafka import KafkaProducer

import json

producer = KafkaProducer(

bootstrap\_servers='kafka-server:port',

security\_protocol='SSL',

ssl\_cafile='/path/to/ca.pem',

ssl\_certfile='/path/to/service.cert',

ssl\_keyfile='/path/to/service.key',

value\_serializer=lambda v: json.dumps(v).encode('utf-8')

)

csv\_file\_path = "/content/drive/MyDrive/Colab Notebooks/etudiants.csv"

with open(csv\_file\_path, mode='r', encoding='utf-8') as file:

csv\_reader = csv.DictReader(file)

for row in csv\_reader:

producer.send('mohamed', value=row)

print(f"Message envoyé : {row}")

producer.flush()

print("Tous les messages ont été envoyés !")

**3.2 Création d'un Consommateur Kafka avec Spark Streaming**

Un consommateur Kafka a été mis en place avec Spark Streaming pour recevoir les données en temps réel. Ces données ont été traitées de la même manière que les données par lots dans Spark.

python

Copier le code

from kafka import KafkaConsumer

import json

consumer = KafkaConsumer(

'mohamed',

bootstrap\_servers='kafka-server:port',

client\_id="CONSUMER\_CLIENT\_ID",

group\_id="CONSUMER\_GROUP\_ID",

security\_protocol='SSL',

ssl\_cafile='/path/to/ca.pem',

ssl\_certfile='/path/to/service.cert',

ssl\_keyfile='/path/to/service.key',

value\_deserializer=lambda m: json.loads(m.decode('utf-8'))

)

for message in consumer:

print(f"Message reçu : {message.value}")

**Partie 4 : Reporting et Analyse**

**4.1 Reporting des Résultats**

Le pipeline ETL a permis de traiter des données en mode batch et en temps réel. Les étapes suivies incluent l'extraction des données, leur transformation (normalisation et filtrage), le calcul de statistiques (moyenne d'âge par filière), et la sauvegarde des résultats. Les résultats ont été enregistrés dans un fichier CSV et envoyés en temps réel à Kafka pour un traitement ultérieur avec Spark Streaming.

Les défis rencontrés incluent des problèmes de connexion avec Kafka (problèmes de métadonnées) qui ont été résolus après quelques tentatives et ajustements.

**4.2 Démonstration du Pipeline**

Le pipeline complet a été testé, y compris l'extraction des données à partir du fichier CSV, leur transformation via Spark, leur envoi à Kafka, leur consommation en temps réel via Spark Streaming, et la sauvegarde des résultats. La démonstration a montré la capacité à traiter et à analyser les données à la fois en mode batch et en temps réel.

**Conclusion**

Cet atelier a permis de démontrer la puissance des outils Apache Spark et Kafka dans la création d'un pipeline de données robuste et flexible. L'intégration du traitement par lots et du traitement en temps réel a permis d'analyser les données de manière complète et d'en tirer des informations utiles pour des décisions rapides et précises.