FL avec Kafka et Spark

Table des matières

1 Introduction et Contexte **2**

2 Architecture du Système **2**

3 Énoncé du Travail **2**

3.1 Phase 1 : Ingestion de flux avec Kafka 2

3.2 Phase 2 : Développement des Nœuds Fog (Spark) 2

3.3 Phase 3 : Agrégation Cloud (FedAvg) 2

4 Technologies et Livrables **3**

1 Introduction et Contexte

Dans l'ère de l'Internet des Objets (IoT), la production massive de données par des capteurs distants pose des défis majeurs en termes de bande passante et de confidentialité. Le **Fog Computing** décentralise le calcul vers la périphérie du réseau, tandis que le **Federated Learning** (Apprentissage Fédéré) permet d'entraîner des modèles d'intelligence artificielle sans centraliser les données brutes.

L'objectif de ce projet est de simuler une infrastructure capable d'entraîner un modèle de détection d'anomalies de manière distribuée en utilisant **Apache Kafka** pour le transport des messages et **Apache Spark** pour le traitement analytique.

2 Architecture du Système

Le projet repose sur une architecture à trois niveaux :

1. **Couche Device (Producteurs) :** Simulateurs de capteurs industriels envoyant des données (vibrations, température) vers Kafka.
2. **Couche Fog (Spark Streaming) :** Nœuds locaux qui effectuent un pré-traitement et un entraînement local du modèle.
3. **Couche Cloud (Agrégateur) :** Un nœud central qui fusionne les paramètres des modèles locaux pour produire un modèle global optimisé.

3 Énoncé du Travail

3.1 Phase 1 : Ingestion de flux avec Kafka

Vous devez configurer un cluster Kafka (ou un broker local) et créer les topics suivants :

— `sensor-data-node-1`, `sensor-data-node-2` : Données brutes.

— `model-weights` : Publication des paramètres du modèle local.

— `global-model` : Diffusion du modèle agrégé.

3.2 Phase 2 : Développement des Nœuds Fog (Spark)

Chaque instance Spark doit :

— Lire les données en temps réel via *Structured Streaming*.

— Appliquer un algorithme de **Stochastic Gradient Descent (SGD)** localement.

— Envoyer les gradients ou les poids calculés vers le topic `model-weights` toutes les N secondes.

3.3 Phase 3 : Agrégation Cloud (FedAvg)

Développer une application Spark (Batch ou Streaming) qui :

— Récupère les poids de tous les nœuds Fog.

— Applique l'algorithme **Federated Averaging** :

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_t^k$$

— Renvoie le nouveau modèle w_{t+1} vers les nœuds pour la prochaine itération.

4 Technologies et Livrables

— **Stack :** PySpark (Spark SQL + MLlib), Kafka Python Client, Docker-Compose.

— **Livrables :**

  — Code source commenté sur GitHub.

  — Un dashboard simple (ou logs) montrant la convergence de la perte (loss) globale.

  — Rapport final expliquant la gestion des échecs d'un nœud Fog.