Rapport de Projet : Détection de Fraude en Temps Réel

Architecture de Streaming Edge–Fog–Cloud

Cheikh Bay Oudaa [C16706]

Département d'Informatique, Master 2 IA, ML & DS
Université de Nouakchott

Encadrant : Dr. El Benany Med Mahmoud

Février 2026

Résumé

Ce rapport présente le développement d'un système distribué de détection de fraude pour le mobile money en Mauritanie. En utilisant une architecture Edge-Fog-Cloud et Apache Kafka, nous démontrons comment traiter des flux de transactions massifs en temps réel tout en réduisant la latence inhérente aux systèmes centralisés.

1 Introduction

Les plateformes de mobile money telles que Bankily et Masrivi jouent un rôle crucial dans l'inclusion financière en Mauritanie. Cependant, la centralisation des données pose des problèmes de scalabilité et de latence. Ce projet propose une solution basée sur le streaming de données, où le traitement est réparti entre la bordure du réseau (Edge) et une couche intermédiaire (Fog) avant d'atteindre le Cloud pour la visualisation.

2 Méthodologie

2.1 Architecture du Système

Le système proposé est structuré en trois couches principales :

Couche Edge – Génération des Transactions

La couche Edge simule les sources de transactions (agences). À l'aide de scripts Python, nous générons des données incluant le montant, l'origine et un indicateur de fraude. Ce processus permet de tester la robustesse du système sans compromettre de vraies données bancaires.

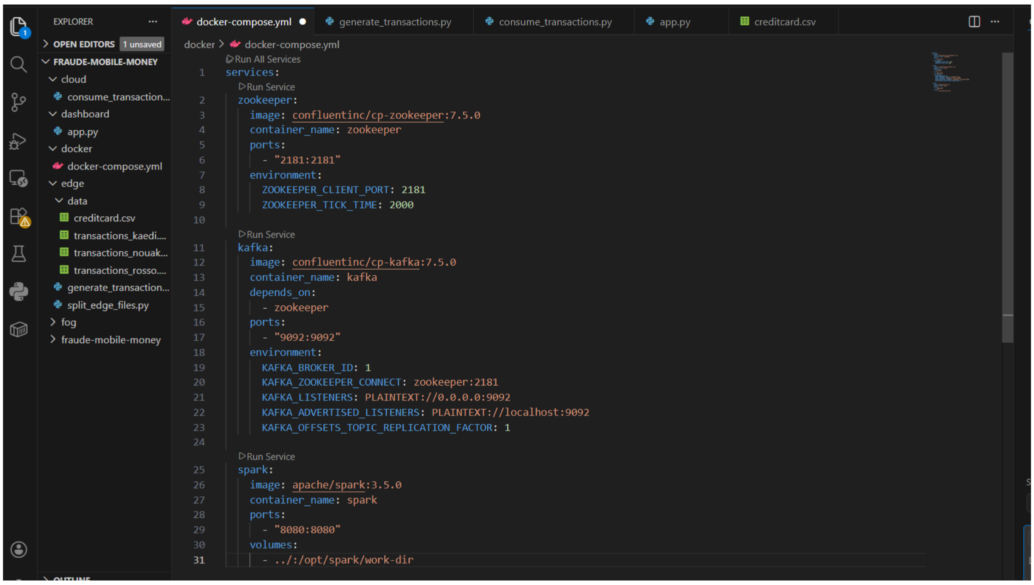


Figure 1 – Figure 1 : Développement des scripts de génération sous VS Code.

Couche Fog – Streaming avec Apache Kafka

Le Fog layer utilise Apache Kafka pour gérer le flux de données. Kafka garantit que même en cas de pic de transactions, les données sont stockées temporairement et transmises sans perte aux consommateurs.

Couche Cloud – Monitoring

La couche finale consomme les messages Kafka pour les analyser. Un tableau de bord Streamlit offre une interface visuelle pour suivre l'état du réseau et les alertes de fraude.

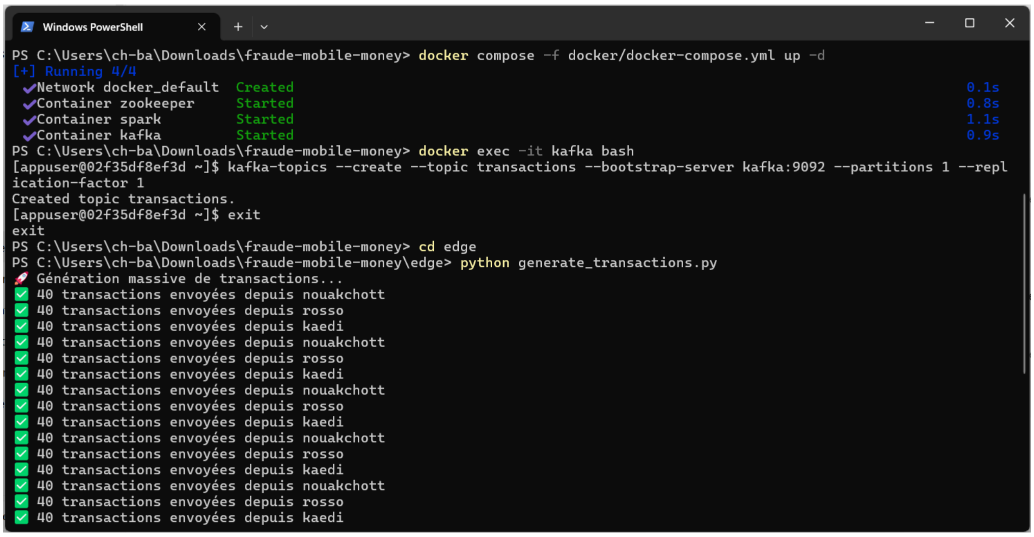


Figure 2 – Figure 2 : Flux de données en temps réel via le broker Kafka.

2.2 Logique d'Étiquetage

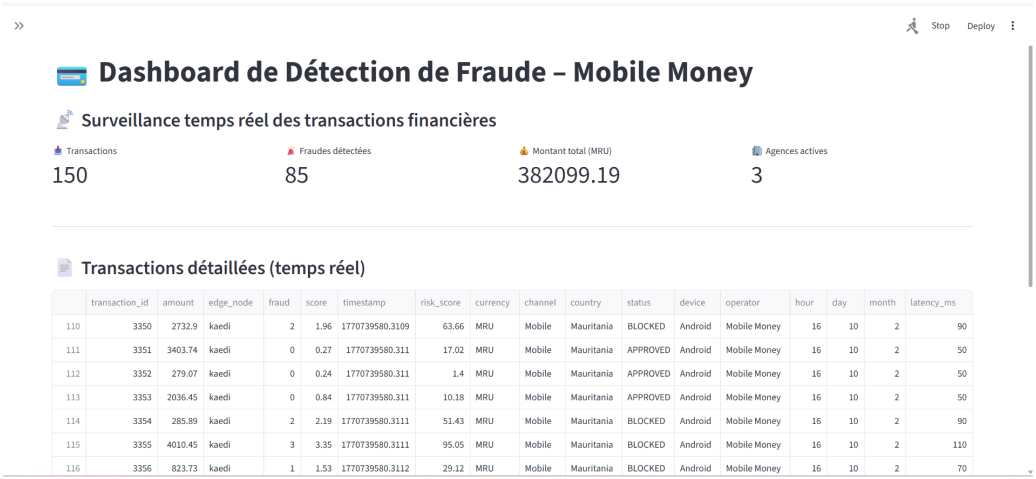
La détection est simulée par une logique probabiliste : la majorité des transactions sont légitimes, et des étiquettes de fraude sont introduites aléatoirement pour tester le pipeline.

3 Configuration Expérimentale

Le déploiement utilise Docker pour Kafka et Zookeeper. L'environnement comprend Python pour la génération, Kafka pour le transport et Streamlit pour l'affichage sous Windows.

4 Résultats et Analyse

Les tests effectués montrent une transmission fluide des données. Le tableau de bord se met à jour instantanément à chaque nouvelle transaction générée à la bordure. Le système dockerisé a prouvé sa stabilité lors de simulations de fortes charges.



	transaction_id	amount	edge_node	fraud	score	timestamp	risk_score	currency	channel	country	status	device	operator	hour	day	month	latency_ms
110	3350	2732.9	kaedi	2	1.96	1770739580.3109	63.66	MRU	Mobile	Mauritania	BLOCKED	Android	Mobile Money	16	10	2	90
111	3351	3403.74	kaedi	0	0.27	1770739580.311	17.02	MRU	Mobile	Mauritania	APPROVED	Android	Mobile Money	16	10	2	50
112	3352	279.07	kaedi	0	0.24	1770739580.311	1.4	MRU	Mobile	Mauritania	APPROVED	Android	Mobile Money	16	10	2	50
113	3353	2036.45	kaedi	0	0.84	1770739580.311	10.18	MRU	Mobile	Mauritania	APPROVED	Android	Mobile Money	16	10	2	50
114	3354	285.89	kaedi	2	2.19	1770739580.3111	51.43	MRU	Mobile	Mauritania	BLOCKED	Android	Mobile Money	16	10	2	90
115	3355	4010.45	kaedi	3	3.35	1770739580.3111	95.05	MRU	Mobile	Mauritania	BLOCKED	Android	Mobile Money	16	10	2	110
116	3356	823.73	kaedi	1	1.53	1770739580.3112	29.12	MRU	Mobile	Mauritania	BLOCKED	Android	Mobile Money	16	10	2	70

FIGURE 3 – Figure 3 : Visualisation des fraudes sur le Dashboard Streamlit.

5 Conclusion

Ce projet valide l'utilisation du streaming distribué pour sécuriser les transactions mobiles. La prochaine étape sera l'intégration de modèles d'intelligence artificielle pour automatiser la détection de motifs frauduleux complexes.

Références

[1] Apache Software Foundation, Documentation Apache Kafka, 2024.

[2] Zaharia et al., “Apache Spark for Big Data Processing,” 2016.

[3] Streamlit Inc., Documentation Streamlit, 2024.