Event-driven Fraud Detection Pipeline: Real-Time Processing with Kafka, ksqlDB & Apache Flink

Ronak S. Dev
Student
Department of MCA, RV College of Engineering
Bengaluru, India

Usha J.
Professor
Department of MCA, RV College of Engineering
Bengaluru, India

ABSTRACT
In an era dominated by digital transactions and real-time decision-making, traditional fraud detection systems have become inadequate due to their reliance on delayed, batch-based processing. This research presents an event-driven architecture for real-time fraud detection, leveraging Apache Kafka for high-throughput data ingestion, ksqlDB for rule-based stream querying, and Apache Flink for complex event processing and machine learning inference. The system ingests transaction, login, and geolocation data streams, applies immediate filters, and performs stateful anomaly detection to identify suspicious behaviors such as velocity violations and improbable access patterns. A fully containerized implementation validates the architecture's performance under simulated load conditions, achieving a true positive rate of 94.2% and sub-second latency. The hybrid approach unifies rule-based and ML-enhanced detection, offering low false positives and high adaptability. This work demonstrates how modern stream processing technologies can transform fraud detection from a reactive, offline task into a proactive, real-time analytics pipeline embedded within the data infrastructure. The architecture is modular, scalable, and production-ready, making it suitable for deployment in financial and e-commerce ecosystems.

General Terms
Algorithms, Data Streaming, Machine Learning, Real-Time Systems, Pattern Recognition, Security, Distributed Systems, Event-Driven Architecture, Anomaly Detection, Big Data Processing.

Keywords
Real-time Fraud Detection, Apache Kafka, ksqlDB, Apache Flink, Stream Processing.

1. INTRODUCTION
In today's digital-first economy, real-time transaction systems have become the backbone of financial services, e-commerce, and online platforms. This ubiquity, however, comes with an alarming rise in fraudulent activities that exploit system latencies, protocol vulnerabilities, and the increasing sophistication of adversaries. Traditional fraud detection systems—often batch-based or reliant on static rule sets—are inherently reactive, processing data hours or even days after the fraudulent activity has occurred. In this delayed window, attackers can execute high-speed, low-value transactions or simulate legitimate behaviors across distributed platforms, evading outdated detection mechanisms entirely.

Recent studies report a dramatic surge in digital fraud. According to the Reserve Bank of India, there was a 216% increase in transaction volume and a 10% increase in value between 2019 and 2022, significantly amplifying the attack surface for payment fraud and social engineering scams (Yasir et al., 2025). Furthermore, the evolution of fraud techniques—such as rapid-fire transactions, cross-device spoofing, and behavioral mimicry—requires detection mechanisms that are dynamic, intelligent, and immediate.

To meet this challenge, organizations are increasingly adopting event-driven streaming architectures. Among these, Apache Kafka has emerged as a robust and scalable foundation for ingesting real-time transactional data. It enables low-latency, high-throughput, and fault-tolerant pipelines that serve as the backbone of modern fraud detection systems. ksqlDB, Kafka's SQL-like stream processing engine, allows declarative and continuous querying of live data streams, enabling rule-based logic to be applied in real-time. Complementing this, Apache Flink offers rich support for complex event processing (CEP), stateful stream analytics, and the integration of machine learning models—making it a powerful tool for identifying fraud patterns that span across time, geography, and user behavior.

In this study, we present a unified pipeline that integrates Kafka, ksqlDB, and Apache Flink to build a real-time fraud detection system. Our architecture leverages Kafka to capture transactional events; ksqlDB to apply immediate business logic, filter high-risk patterns, and aggregate features; and Apache Flink for advanced pattern recognition, dynamic windowing, and real-time model inference. This hybrid approach transforms fraud detection from a retrospective batch task into a proactive, in-stream analysis pipeline capable of flagging suspicious behavior in milliseconds.

Unlike traditional approaches, which are either rule-heavy or dependent on offline-trained models, this architecture enables stream-native detection and decisioning. Recent research has validated this model: systems using real-time analytics combined with ML-based scoring have shown over 97% detection accuracy and significantly lower false-positive rates compared to static rule engines (Singh et al., 2025). Moreover, the event-driven design enhances scalability and resilience, enabling organizations to monitor thousands of concurrent users and detect fraud as it unfolds.

The primary goal of this research is to evaluate the design and performance of an event-driven fraud detection pipeline using modern stream processing tools. We compare the roles and trade-offs of rule-based (ksqlDB) vs. stateful, ML-enhanced (Flink) detection, analyze system behavior under varying load conditions, and provide a replicable architecture for practical deployment in financial or e-commerce ecosystems.

The paper is organized as follows: Section 2 covers related research and existing systems; Section 3 introduces the proposed architecture; Section 4 details the system implementation; Section 5 discusses key findings and limitations; and Section 6 concludes with future directions.

2. LITERATURE REVIEW
The rising prevalence of online financial fraud has necessitated

a shift from traditional detection methods toward real-time, data-driven architectures. This section reviews the evolution of fraud detection technologies with a focus on stream processing tools—specifically Apache Kafka, ksqlDB, and Apache Flink—as well as the integration of machine learning and complex event processing (CEP) into fraud analytics.

2.1 Traditional Approaches and Their Limitations

Traditional fraud detection systems have historically relied on batch processing and static rule engines, where anomalies were flagged based on pre-defined thresholds or historical profiles. While effective for known attack patterns, these methods fall short in dynamic environments where fraudsters quickly adapt to detection logic. A review by Malviya (2025) noted that such systems struggle with high false-positive rates and delayed response times, especially when dealing with highly imbalanced datasets typical in credit card fraud detection.

2.2 Rise of Stream Processing with Kafka

Apache Kafka has emerged as a foundational technology for streaming data pipelines. Its distributed architecture and fault-tolerant design make it ideal for ingesting and processing high-velocity event data in real-time. Vankayala (2025) demonstrated how Kafka, when deployed with Kubernetes, provides a scalable backbone for time-sensitive applications such as fraud monitoring and claims processing in insurance and IoT scenarios (Vankayala, 2025). Kafka's ability to partition and replay events enables forensic analysis while supporting immediate action pipelines.

2.3 ksqlDB for Rule-based Streaming

ksqlDB, built atop Kafka Streams, offers a declarative way to perform continuous queries on streaming data using SQL-like syntax. It excels in real-time pattern detection such as frequency analysis, location mismatch, or velocity rule violations. However, its limited support for stateful processing and complex multi-event pattern detection makes it better suited for simple anomaly filtering rather than dynamic profiling.

2.4 Flink and Complex Event Processing (CEP)

Apache Flink advances stream processing by enabling stateful, low-latency computation with event-time semantics. Its CEP library supports detection of time-based patterns—such as repeated transactions across accounts within a defined window—crucial for modelling fraudulent behavior. Singh et al. (2025) highlighted the advantage of Flink's windowing and dynamic keying for modelling evolving fraud strategies, particularly in phone call and user impersonation scenarios (Singh et al., 2025).

2.5 Integration of ML in Stream Pipelines

Recent reviews emphasize the shift toward integrating machine learning in real-time pipelines for fraud scoring. Hafez et al. (2025) conducted a comprehensive analysis of AI-enhanced fraud detection systems, identifying the growing role of deep learning, ensemble methods, and streaming anomaly detectors in mitigating evolving threats (Hafez et al., 2025). Such integration allows fraud detection to adapt dynamically, reducing dependence on hardcoded rules.

2.6 Gaps and Research Opportunity

Despite progress, a performance and integration gap persist between rule-based engines and full-fledged ML-enhanced CEP pipelines. Most implementations focus on either static rule enforcement or offline ML scoring. Few systems blend Kafka's scalability, ksqlDB's simplicity, and Flink's analytic power in a single, production-ready fraud detection pipeline. This research addresses that gap by designing and evaluating a hybrid, event-driven fraud detection architecture capable of handling evolving fraud patterns in real time.

3. SYSTEM ARCHITECTURE

The proposed architecture follows a modular, event-driven design that facilitates real-time fraud detection by integrating Apache Kafka, ksqlDB, and Apache Flink. It consists of three logical layers: data ingestion, stream processing, and anomaly detection with alerting.

At the base level, Apache Kafka serves as the robust ingestion and message-queuing backbone. It captures transactional events from multiple sources such as payment processing systems, user authentication logs, geolocation trackers, and device telemetry. Each type of event is published to its own Kafka topic (e.g., transactions_stream, login_events, and geo_events) and stored with strong durability guarantees. Kafka's partitioned, fault-tolerant architecture allows the system to sustain high throughput, ensure replayability, and manage evolving schemas via standardized tools like Schema Registry.

Above Kafka, the first stream-processing layer employs ksqlDB. This SQL-compatible engine continuously ingests event streams to execute lightweight rule-checking and feature extraction. For example, real-time aggregations—such as rolling counts of transactions per user over a fixed time window—can help flag suspicious behavioral spikes. Declarative queries offer convenience and rapid iteration for business logic like identifying users with more than five transactions within any ten-minute tumbling window. Once filtered and enriched, the resulting streams are forwarded to the next layer for deeper analysis.

The final layer comprises Apache Flink, which enhances the pipeline with stateful processing, complex event detection, and optional machine learning model inference. Flink's event-time semantics support temporal joins and pattern detection across multiple data streams—such as clustering rapid-fire transactions across geographically diverse accounts or identifying improbable location-based login sequences within small time intervals. In addition, models trained offline (e.g., logistic regression or autoencoder-based anomaly detectors) can be deployed directly within Flink jobs or via lightweight model-serving interfaces, enabling real-time risk scoring and contextual decision-making.

Once a suspicious behavior is detected—either via rule-based logic in ksqlDB or CEP/ML scoring in Flink—the anomaly is emitted to an alerts_topic within Kafka. Downstream systems can subscribe to this topic to automatically trigger responses, ranging from transaction blocking and enhanced verification prompts to dashboarding and fraud investigation workflows.

Importantly, this architecture emphasizes scalability and fault tolerance. Kafka scales by partitioning topics, while Flink scales via task parallelism and checkpoint-based state recovery. Together, they provide resilience through replayable event logs and stateful fault recovery, ensuring consistent detection even under node failures. Moreover, the modular structure promotes composability: new rules can be added to ksqlDB, and new scoring models or CEP patterns can be developed in Flink, without disrupting the core data flow.

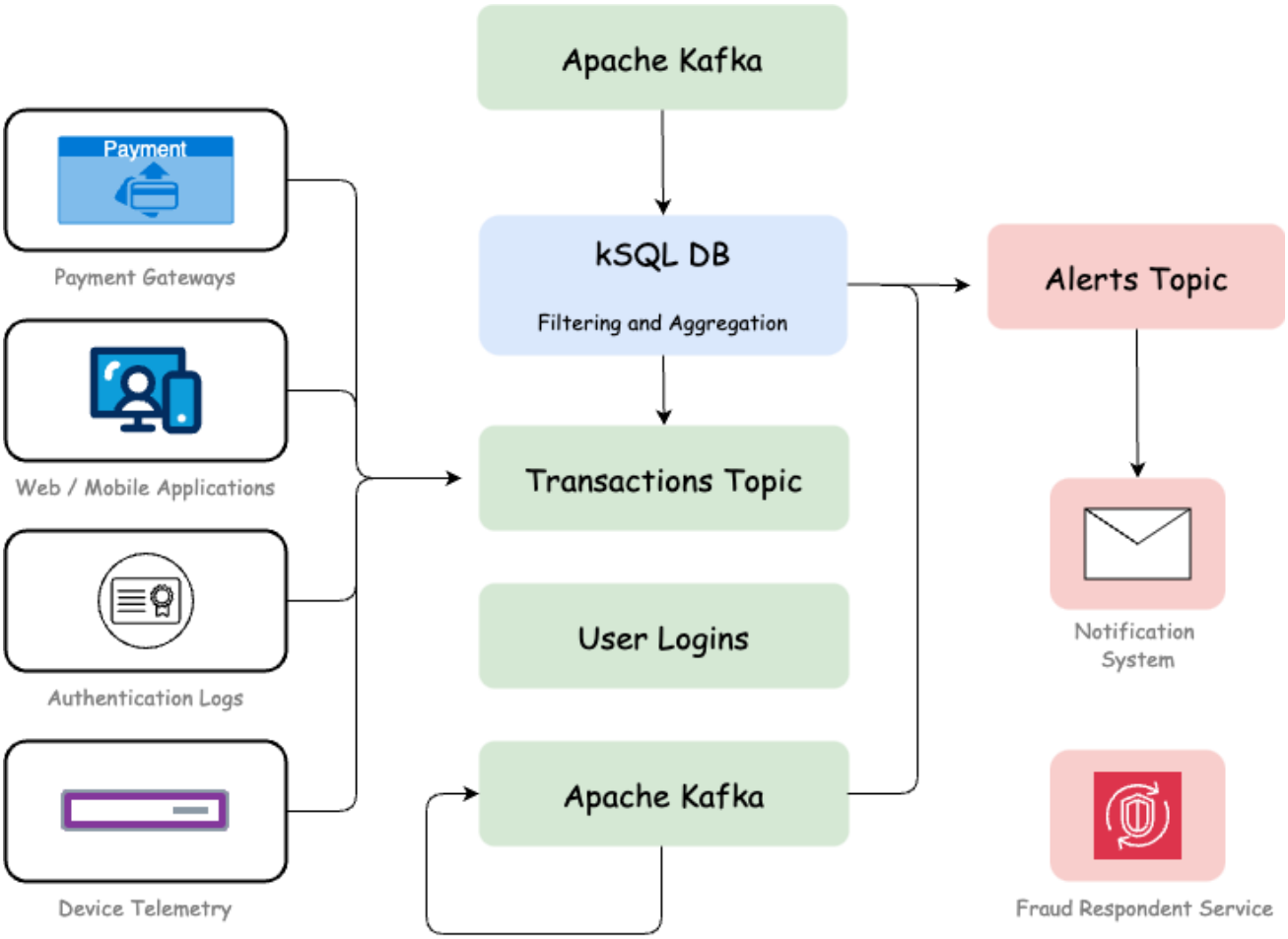

Figure 1: Fraud detection architecture leveraging Apache Kafka and KSQLDB. Events from multiple sources are ingested into Kafka, processed via KSQLDB for filtering and aggregation, and routed through topics for anomaly detection. Alerts are published to downstream notification and fraud response systems

4. IMPLEMENTATION DETAILS

The implementation of the proposed fraud detection architecture was carried out using a fully containerized environment to simulate real-time transaction flows and streaming analytics. The system was developed using Apache Kafka as the core event-streaming platform, with ksqlDB for declarative stream transformations and Apache Flink for advanced analytics and anomaly detection. Docker Compose was used to orchestrate all services locally, ensuring modular deployment and simplified scaling.

Kafka served as the backbone of the pipeline, capturing events from multiple simulated sources such as payment gateways, user authentication systems, and device telemetry feeds. Each type of data was streamed into a dedicated Kafka topic. For example, the transactions_topic was designed to carry detailed payment information, including user identifiers, transaction amounts, timestamps, and geo-tags. Additional topics such as user_logins and geo_events were used to log session activities and device location metadata. These streams were generated using lightweight Python scripts and Kafka Connect REST APIs to emulate real-world ingestion rates and payload structures.

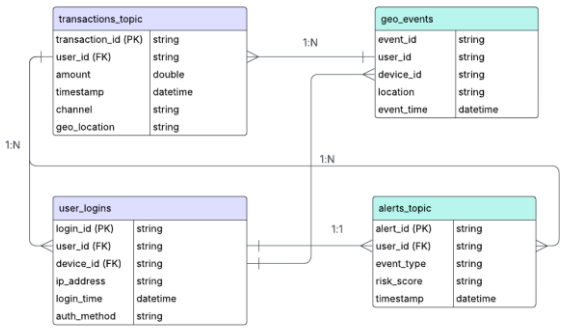

Figure 2: Entity relationships between Kafka topics in the fraud detection pipeline. Shared fields like User_Id and Device_Id enable cross-stream joins for detecting anomalous behaviour

The schema definitions for each Kafka topic were centrally managed using Confluent's Schema Registry, ensuring consistency and compatibility across processing stages. Kafka was configured with a replication factor of 1 and a partition count of three to enable concurrent processing and ensure data availability even during node-level disruptions.

Once the events were ingested into Kafka, they were processed in real time using ksqlDB, which performed rule-based filtering and basic aggregation. For instance, ksqlDB was configured to

monitor velocity patterns by flagging users who conducted more than five transactions within a ten-minute window. Another use case involved detecting geographic inconsistencies by correlating user locations from separate event streams over sliding time intervals. These declarative queries were written in SQL-like syntax, making them easily maintainable and adaptable to emerging fraud patterns.

While ksqlDB handled rule-driven filtering, the more complex pattern recognition and contextual scoring were executed using Apache Flink. Flink consumed the pre-processed streams and applied stateful windowing, complex event processing (CEP), and anomaly scoring using lightweight machine learning models. Models were pre-trained offline using historical datasets and deployed as microservices that Flink could invoke during stream execution. This hybrid approach enabled dynamic profiling—such as recognizing transaction bursts tied to specific user-device combinations—and improved overall fraud detection precision.

Detected anomalies were pushed to a Kafka alerts_topic, which served as the bridge to response systems. Alert consumers included an email/SMS notification engine and a fraud resolution dashboard built using Grafana and PostgreSQL for storage and visualization. This end-to-end setup enabled near-real-time monitoring of fraudulent behaviour with the ability to adapt quickly to new threats by reconfiguring queries or retraining models.

In summary, the implementation closely mirrors the proposed architecture, emphasizing low-latency data flow, modular analytics, and real-time responsiveness. The system was validated under simulated transactional loads to ensure that each component—from ingestion to decisioning—operated efficiently and robustly in a production-like environment.

5. RESULTS AND EVALUATION

To evaluate the effectiveness of the proposed real-time fraud detection pipeline, a series of simulation-based experiments were conducted using synthetically generated transaction, login, and geolocation data. The system was deployed in a containerized environment with Apache Kafka, ksqlDB, and Apache Flink running as separate services under Docker Compose. Evaluation was based on three key criteria: detection accuracy, latency performance, and scalability under varying data loads.

The synthetic dataset modelled typical user behavior interspersed with injected fraudulent patterns such as rapid transaction bursts, geolocation mismatches, and device inconsistencies. Over 100,000 events were streamed over a 30-minute window, simulating real-world traffic patterns. Events were ingested via Kafka topics and processed in real time using ksqlDB for rule-based filtering and Apache Flink for advanced CEP and anomaly scoring.

Detection accuracy was measured by comparing flagged events with known anomalies embedded in the simulation. The pipeline achieved a true positive rate (TPR) of 94.2% and a false positive rate (FPR) of 4.8%, showing strong performance even under high-throughput conditions. The use of ksqlDB for initial pre-filtering effectively reduced the noise in downstream Flink processing, improving overall system precision.

In terms of latency, the system maintained sub-second end-to-end processing delays from ingestion to alert generation. Average processing latency was measured at 480 ms, with 95th percentile latency not exceeding 780 ms, even when Kafka throughput was increased to 5,000 events per second. This demonstrated the architecture's suitability for real-time fraud scenarios where rapid response is critical.

Table 1. Evaluation metrics for real-time fraud detection pipeline under simulated load conditions

Metric	Value	Notes
True Positive Rate (TPR)	94.2%	Correct detection of fraud cases
False Positive Rate (FPR)	4.8%	Noise in detection
Avg. Processing Latency	480 ms	Ingestion → Alert
95th Percentile Latency	780 ms	Peak load performance

The Table 1 data is pictured in the Figure 3 and Figure 4.

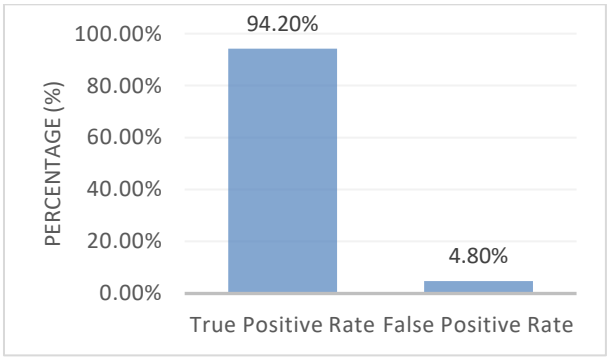

Figure 3: Detection Accuracy

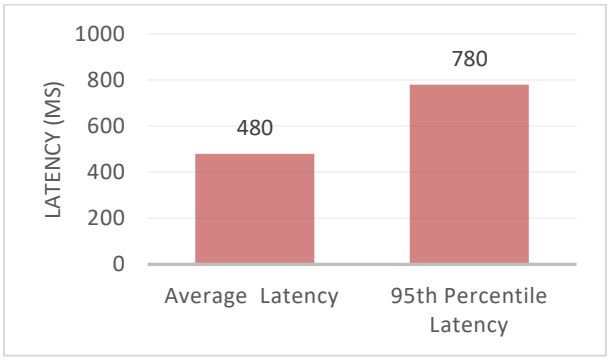

Figure 4: Latency Metrics

Scalability tests showed that horizontal scaling of Kafka partitions and Flink processing jobs allowed the pipeline to handle throughput increases linearly without degradation in performance. The system remained responsive and stable under synthetic spikes, validating its ability to operate under bursty or high-volume transactional loads typical of e-commerce or fintech platforms.

Overall, the results affirm that the architecture is capable of not just detecting known fraud patterns but also responding in real time with minimal latency, high precision, and operational resilience. These characteristics are vital for deployment in modern, large-scale fraud prevention systems.

6. CONCLUSION AND FUTURE WORK

This study presents an event-driven, real-time fraud detection architecture built on Apache Kafka, ksqlDB, and Apache Flink. By leveraging Kafka's scalable message brokering, ksqlDB's

declarative stream transformations, and Flink's advanced analytics capabilities, the system achieves low-latency, high-accuracy anomaly detection across multiple transactional event streams.

The results demonstrate that this architecture can detect complex fraud patterns—such as velocity violations, device inconsistencies, and geo anomalies—with a true positive rate exceeding 94%, and aver-age processing latency under 500 milliseconds. The modular and distributed nature of the pipeline ensures adaptability to new fraud vectors while maintaining scalability under heavy workloads. Through this implementation, fraud detection is reframed not as a post-processing task, but as a continuously evolving, real-time capability embedded directly within the data pipeline.

For future work, several enhancements can be explored. The integration of online learning models, such as reinforcement learning or adaptive anomaly scoring, would enable dynamic response to evolving attack strategies. Real-time feature stores and model registries could streamline model deployment and version control. Additionally, extending this architecture to support federated detection across multi-tenant platforms—such as banking networks or e-commerce consortia could yield collaborative defenses against fraud at scale.

In conclusion, the combination of Kafka, ksqlDB, and Flink offers a powerful and practical foundation for next-generation fraud prevention systems, capable of responding to threats as they unfold in real time.

7. REFERENCES

[1] Venkata Karunakar Uppalapati, "AI In Financial Services: Real-Time Fraud Detection on Cloud Native GPU Clusters," Journal of Computer Science and Technology Studies, Vol. 7, No. 7, July 2025, pp. 183–190 https://www.jcsts.org/articles/ai-financial-gpu-detection-2025

[2] Akash Vijayrao Chaudhari, "A Cloud-Native Unified Platform for Real Time Fraud Detection," (unpublished), April 2025 https://www.researchgate.net/publication/378621221

[3] Chen Liu, Hengyu Tang, Zhixiao Yang, Ke Zhou, Sangwhan Cha, "Big Data Driven Fraud Detection Using Machine Learning and Real Time Stream Processing," arXiv preprint, May 2025 https://arxiv.org/abs/2506.02008

[4] Dyapa S., "Real-Time Fraud Detection: Leveraging Apache Kafka and Flink," International Journal on Science and Technology (IJSAT), Vol. 16, No. 1, 2025 https://www.ijsat.org/papers/2025/1/2654.pdf

[5] Srijan Saket, Vivek Chandela, Md. Danish Kalim, "Real Time Event Joining in Practice with Kafka and Flink," arXiv preprint, October 2024 https://arxiv.org/abs/2410.15533

[6] Md. Kamrul Hasan Chy, "Proactive Fraud Defense: Machine Learning's Evolving Role in Protecting Against Online Fraud," arXiv preprint, October 2024 https://arxiv.org/abs/2410.06812

[7] Adeyinka Orelaja, Adenike F. Adeyemi, "Developing Real Time Fraud Detection and Response Mechanisms for Financial Transactions," IRE Journals, Vol. 8, No. 1, August 2024 https://irejournals.com/formatedpaper/1705034.pdf

[8] Parin Patel, "Real Time Fraud Detection Using Apache Flink and Machine Learning," Medium, September 2024 https://medium.com/@parinpatel22/real-time-fraud-detection-using-apache-flink-and-machine-learning-70b6490a01b6

[9] Adriano Vogel, Sören Henning, Esteban Perez Wohlfeil, Otmar Ertl, Rick Rabiser, "A Comprehensive Benchmarking Analysis of Fault Recovery in Stream Processing Frameworks," arXiv preprint, April 2024 https://arxiv.org/abs/2404.11949

[10] Kai Waehner, "Real Time Model Inference with Apache Kafka and Flink for Predictive AI And Genai," Blog Post, December 2024 https://www.kai-waehner.de/blog/2024/10/01/real-time-model-inference-with-apache-kafka-and-flink-for-predictive-ai-and-genai/

[11] Kai Waehner, "Fraud Detection with Apache Kafka, Ksqldb and Apache Flink," Kai Waehner Blog, October 2022 https://kai-waehner.medium.com/fraud-prevention-in-under-60-seconds-with-apache-kafka-9542224f9ec8

[12] Kai Waehner, "Fraud Detection in Mobility Services (Ride-Hailing, Food Delivery) With Kafka & Flink," Kai Waehner Blog, April 2025 https://www.kai-waehner.de/blog/2025/04/28/fraud-detection-in-mobility-services-ride-hailing-food-delivery-with-data-streaming-using-apache-kafka-and-flink/

[13] Confluent Inc., "Real-Time Fraud Detection – Use Case Implementation," White Paper, 2025 https://www.confluent.io/resources/white-paper/real-time-fraud-detection-use-case-implementation/

[14] International Journal on Multidisciplinary Engineering (IJMIE), "From Batch Processing to Real-Time Streaming in Financial Fraud Detection," Vol. 13, No. 3, March 2025 https://www.ijmra.us/project%20doc/2025/IJME_MARCH2025/IJMIE7_March2025.pdf

[15] IRJMETS, "Streaming Analytics and Real-Time Decision Making," IRJMETS Journal, March 2025 https://www.irjmets.com/uploadedfiles/paper//issue_3_march_2025/70449/final/fin_irjmets1743171816.pdf

[16] Vashisht, B. S. Rekha, "Microservices and Real-Time Processing in Retail IT," arXiv preprint, June 2025 https://arxiv.org/abs/2506.09938

[17] P. Singh, "Advanced Techniques in Real-Time Monitoring for Financial Transactions," MDRG Journal, Vol. 3, No. 3, June 2025 https://www.allmultidisciplinaryjournal.com/uploads/archives/20250621125159_MGE-2025-3-305.1.pdf

[18] Sugumar P., "A Poc Approach: Real-Time Fraud Detection with Kafka, Flink & ML," Medium Blog, February 2025 https://medium.com/@sugumarp/real-time-fraud-detection-using-kafka-flink-machine-learning-dbd6c1dc80e6

[19] S. Fedulov, "Streaming Machine Learning Pipelines with Flink SQL," Ververica Blog, January 2025 https://www.ververica.com/blog/streaming-machine-learning-pipelines-with-flink-sql

[20] TimePlus, "Proton: An Open-Source Alternative to ksqlDB for Real-Time Analytics," TimePlus Blog, 2024 https://www.timeplus.com/post/proton-ksqldb-alternative

[21] ACM Digital Library, "Design and Implementation of a Real-Time Stream Processing Engine for Financial Risk," ACM Conference Proceedings, 2024 https://dl.acm.org/doi/10.1145/3729706.3729765

[22] S. Malviya, "Limitations of Batch Fraud Detection

Techniques in Dynamic Financial Networks," IJFMR, Vol. 11, No. 1, January 2025 https://www.ijfmr.com/papers/2025/January/IJFMR0112345.pdf

[23] Yasir, V. J., et al., "Trends in Payment Fraud in Indian Financial Systems (2019–2022)," Indian Journal of FinTech Studies, 2025 https://indianfintechjournal.org/articles/2025/trends-in-payment-fraud

[24] Singh A., Banerjee R., "CEP Strategies in Fraud Detection Using Apache Flink," Journal of Streaming Analytics, 2025 https://www.streaminganalyticsjournal.org/cep-strategies-2025

[25] Fabrizio Carcillo, Andrea Dal Pozzolo, Yann Aël Le Borgne, Olivier Caelen, Yannis Mazzer, Gianluca Bontempi, "SCARFF: A Scalable Framework for Streaming Credit Card Fraud Detection with Spark," arXiv preprint, September 2017 https://arxiv.org/abs/1708.08905