ISSN: 2583-9888
DOI: doi.org/10.51219/JAIMLD/ravi-kiran-alluri/590

URF PUBLISHERS
connect with research world

Journal of Artificial Intelligence, Machine Learning and Data Science

https://urfpublishers.com/journal/artificial-intelligence

Vol: 1 & Iss: 1

Research Article

Building Self-Healing Fraud Detection Pipelines in Cloud-Native Environments

Ravi Kiran Alluri*

Citation: Alluri RK. Building Self-Healing Fraud Detection Pipelines in Cloud-Native Environments. *J Artif Intell Mach Learn & Data Sci 2023* 1(1), 2826-2831. DOI: doi.org/10.51219/JAIMLD/ravi-kiran-alluri/590

Received: 03 January, 2023; **Accepted:** 28 January, 2023; **Published:** 30 January, 2023

***Corresponding author:** Ravi Kiran Alluri, USA, E-mail: ravikiran.alluirs@gmail.com

ABSTRACT

The risk and complexity of fraud have skyrocketed due to the widespread use of digital transactions in banking, financial services, and e-commerce. The design, implementation, and maintenance of data pipelines have changed concurrently with the rise of cloud-native architectures, which are distinguished by microservices, containerization, orchestration platforms like Kubernetes, and elastic scalability. However, these dynamic, dispersed, and failure-prone environments are not naturally suited for operating conventional fraud detection systems. These legacy solutions are not appropriate for cloud-native deployments that require continuous availability and autonomous fault recovery because they frequently have delayed detection, lack resilience, and significant downtime in the event of component failures.

A novel framework for building self-healing fraud detection pipelines in cloud-native environments is presented in this paper. A self-healing pipeline system can automatically detect and initiate corrective actions without human intervention. Examples of these faults include service crashes, model degradation, anomalies in data ingestion, and latency bottlenecks. Several cutting-edge cloud-native technologies and design paradigms are integrated into our suggested architecture. Fundamentally, the pipeline uses containerized services spread throughout a Kubernetes cluster to guarantee automated failover, scalability, and isolation. Apache Kafka makes real-time event streaming and component decoupling possible, and machine learning models used for fraud classification can be deployed and updated using cloud-native model hosting platforms like Azure ML, AWS SageMaker, or Google Vertex AI.

The layered approach achieves key self-healing capabilities: Kubernetes-native operators that perform rollback, redeployment, or traffic redirection based on predefined policies; sidecar-based observability agents that identify abnormalities in service behavior; and health monitoring through Prometheus and Grafana. Using past transaction data, we also incorporate an intelligent model monitoring component to detect concept drift and retrain workflows. This guarantees that fraud detection models will perform well in prediction, even as fraud patterns change over time.

Our implementation shows that the system retains accuracy and operational continuity even in partial node outages, message queue overload, and corrupted data segments. Metrics including mean time to detect (MTTD), mean time to recovery (MTTR), fraud detection accuracy, and overall system throughput are assessed by testing the pipeline under synthetic and real-world scenarios. Comparative findings show a notable improvement over non-resilient architectures, particularly when upholding service-level objectives (SLOs) during disruptions.

This research provides the reference architecture and empirical validation for self-healing fraud detection in cloud-native infrastructures, which adds to the developing field of resilient AI systems. Rapid integration with DevSecOps pipelines and observability tools is made possible by the modular design, making extensibility and regulatory compliance easier. Future developments will concentrate on adding remediation based on reinforcement learning, enabling decentralized trust layers, and expanding the framework to multi-cloud and hybrid environments. Fraud prevention teams, cloud architects, and compliance officers who want to operationalize AI-driven security analytics at scale while guaranteeing high availability, auditability, and low downtime should take note of this work.

Keywords: Self-healing systems, fraud detection pipelines, cloud-native architecture, microservices, Kubernetes, event-driven architecture, anomaly detection, observability, real-time analytics, fault tolerance, model drift detection, financial technology, machine learning operations (MLOps), autonomous recovery, compliance-aware computing

1. Introduction

The rapid digitization of financial services, e-commerce platforms, and digital banking has led to an exponential surge in online transactions. Alongside these advancements, fraudulent activities have escalated in volume and sophistication. Often designed for on-premise infrastructure or monolithic architectures, traditional fraud detection systems have become increasingly inadequate in combating modern fraud threats. These systems typically lack the agility, scalability, and resilience required to handle the dynamic nature of contemporary cloud-native environments. Additionally, operational dependencies, rigid data pipelines, and a lack of self-recovery mechanisms render them prone to disruptions, particularly in service outages, infrastructure drift, or evolving fraud patterns.

Cloud native computing has emerged as a transformative paradigm that enables scalable, modular, and automated service deployment. Leveraging microservices, container orchestration platforms like Kubernetes, serverless functions, and event-driven frameworks, organizations can develop and deploy highly responsive and loosely coupled systems. These characteristics are ideal for data-intensive applications such as real-time fraud detection. However, the inherent volatility of distributed cloud-native systems poses significant reliability and monitoring challenges. Services may fail, scale unpredictably, or experience data inconsistencies that can degrade fraud detection accuracy or delay decision-making. To address these challenges, there is a critical need for fraud detection systems that can autonomously monitor, detect, and repair faults in real time hence, the rise of self-healing architectures.

A self-healing fraud detection pipeline is an intelligent system designed to manage runtime failures autonomously, ensure minimal disruption in data processing workflows, and maintain model performance under volatile conditions. This capability is not merely about restarting failed services; it involves deeper integration of telemetry, observability, and automated decision logic, creating a closed-loop feedback system capable of handling anomalies across the pipeline. These may include data ingestion faults, service crashes, concept drift in ML models, degraded latency in prediction APIs, or deviations in expected transaction behavior.

This paper proposes and evaluates a comprehensive framework for implementing such a self-healing fraud detection system within a cloud-native infrastructure. We employ Kubernetes for orchestrating containerized microservices and ensuring fault isolation, Apache Kafka for event-streaming between loosely coupled components, and Prometheus for real-time monitoring. Model serving is handled through managed cloud AI platforms, which are monitored and retrained based on drift detection and feedback loops. Furthermore, the system utilizes circuit breaker patterns, auto-scaling, and rollback mechanisms to ensure continuous uptime and consistent data delivery. By embedding policy-driven self-repair mechanisms and system health validators, the architecture not only recovers from faults but also prevents their recurrence through predictive monitoring and dynamic adjustment.

This research is significant because it can potentially reduce operational costs associated with manual incident resolution, improve the precision of fraud detection models under adverse conditions, and support regulatory compliance by maintaining full audit trails of system behavior. The proposed system demonstrates its ability to achieve low mean time to recovery (MTTR), high fraud detection recall, and sustained throughput in real-time data environments through extensive experimental validation across different failure scenarios and fraud detection workloads.

2. Literature Review

The challenges of fraud detection have been extensively addressed in the literature, particularly in domains such as e-banking, digital commerce, and insurance claims processing. Early fraud detection systems primarily relied on rule-based mechanisms, where domain experts codified suspicious behavioral patterns into static rules [1]. These systems, though interpretable, were inflexible and easily circumvented by evolving fraud strategies. With the advent of machine learning, more adaptive systems were introduced that could learn patterns from historical transaction data. These approaches leveraged supervised models such as logistic regression, decision trees, and neural networks, which improved fraud detection accuracy but introduced new challenges related to data drift and real-time model degradation [2,3].

The emergence of big data and streaming analytics introduced further innovations in the field. Systems like Apache Flink and Kafka Streams facilitated low-latency fraud detection by enabling real-time feature extraction and continuous scoring pipelines [4]. Yet, these architectures largely depended on static configurations and lacked resilience in infrastructure failures or pipeline bottlenecks. Cloud-native technologies were adopted to address scalability and agility concerns, with microservices and container orchestration frameworks like Kubernetes becoming standard tools for deploying distributed data pipelines [5]. Despite these advances, most fraud detection pipelines still lacked built-in mechanisms for self-repair or intelligent fault management, often requiring manual intervention during failures.

The concept of self-healing systems gained traction in parallel within the domain of cloud-native infrastructure. Research in autonomic computing proposed architectural patterns where systems could sense environmental changes, analyze anomalies, and act to correct or mitigate issues without human oversight [6]. These principles have been integrated into container orchestration platforms, with Kubernetes providing primitives for health checks, auto-scaling, and service restart policies. More sophisticated frameworks incorporate service mesh architectures (e.g., Istio) and observability stacks (e.g., Prometheus, Grafana, and Loki) to monitor application behavior and trigger recovery workflows [7,8].

When applied to machine learning operations (MLOps), self-healing has gained new meaning. Studies have highlighted the importance of model monitoring, concept drift detection, and automated retraining as key components in ensuring robust AI deployments [9]. For example, researchers have developed frameworks that use statistical divergence measures and real-time feedback loops to monitor model drift and initiate retraining jobs on demand [10]. However, integrating these mechanisms into end-to-end fraud detection pipelines remains an open challenge, particularly when deployed at scale in cloud-native environments. There is limited literature combining self-healing infrastructure with fraud detection logic, which represents a critical gap that this paper seeks to address.

A few efforts have begun to emerge to bridge this divide. Gupta et al. proposed an adaptive fraud detection model embedded within a microservices framework, but the self-healing aspects were limited to basic container restarts [11]. Similarly, Kundu and Sharma implemented a cloud-based fraud detection service with alerting capabilities, yet lacked feedback-driven model correction or automated pipeline recovery [12]. Meanwhile, advances in event-driven architectures and serverless platforms have opened new avenues for reactive fraud analytics, though these platforms still lack native support for intelligent error diagnosis and mitigation [13].

The study builds upon resilient microservice architectures, real-time ML model governance, and autonomous fault-tolerant computing principles. By fusing these threads into a single architecture, this work presents a unified, scalable, and self-healing pipeline capable of detecting fraud in real time and responding autonomously to faults, model decay, and infrastructure volatility. The proposed system thus addresses the dual challenges of detection accuracy and operational resilience, making it suitable for deployment in highly regulated and mission-critical financial environments.

3. Methodology

Developing a self-healing fraud detection pipeline in cloud-native environments requires a multi-layered strategy that combines intelligent anomaly detection, real-time data streaming, containerized microservices, and autonomous remediation mechanisms. The design prioritizes continuous availability, little manual intervention, and adaptive response to algorithmic and infrastructure failures. Cloud-native operational models and dynamic, feedback-driven fraud analytics are combined at the heart of this architecture to enable the system to detect disruptions, assess their effects, and independently carry out recovery logic in almost real time.

As the backbone for microservice deployment, scaling, and fault isolation, Kubernetes orchestrates the pipeline's foundational layer. Data ingestion, feature engineering, model inference, post-processing, and logging are all functional pipeline components deployed as separate, loosely coupled services within Docker containers. Kubernetes's built-in features provide the baseline resilience capabilities, which include rolling updates, readiness probes, liveness probes, and pod auto-restart. Custom health-check sidecars are injected alongside essential services to monitor domain-specific behaviors such as data volume throughput, latency spikes, and prediction error rates.

The system uses Apache Kafka as a distributed event-streaming platform for decoupled service interaction and real-time ingestion. Kafka topics are designed for alert generation, fraud scores, feature transformation, and raw transaction ingestion. This event-driven model ensures that systemic unavailability or data loss doesn't result from temporary failures in downstream services (like model scoring APIs). When service is restored, Kafka consumers can easily recover because they are built to start processing from the last committed offset. Furthermore, as a fallback option if ML models are unavailable or unreliable, Kafka Streams and KSQL are utilized for basic rule-based fraud detection and in-flight feature computation.

TensorFlow Serving or MLflow containerizes the machine learning layer depending on the model type. RESTful APIs expose these inference services, and Kubernetes' Horizontal Pod Autoscaler (HPA) controls autoscaling based on latency and CPU metrics. A Prometheus-Grafana stack is set up to track essential metrics like prediction distribution, inference time, and model drift signals to assess the model's health. The monitoring layer uses Alertmanager to initiate recovery actions through Kubernetes Jobs or serverless functions (like AWS Lambda or Azure Functions) when anomalies are identified, such as notable shifts in data distribution or fraud prediction confidence.

A collection of control loops, implemented as Kubernetes operators, makes up the self-healing orchestration layer. These loops monitor for anomalous conditions and trigger remediation logic. Some examples are reloading a previously stable version of the ML model, restarting services with corrupted memory, or rerouting traffic to a secondary scoring pipeline in a different zone or region. Automated retraining workflows utilizing historical labeled datasets stored in object storage (e.g., S3, Azure Blob) are triggered by more complex failure scenarios, such as extended concept drift. Kubeflow Pipelines and Apache Airflow manage these workflows, enabling dynamic hyperparameter tuning, model evaluation, and redeployment without human intervention.

For robust observability and auditability, the system records all actions and failure events in a centralized logging platform, such as OpenSearch or Elasticsearch with Kibana (ELK stack), which facilitates compliance reporting and forensic analysis. All layers enforce compliance with data protection standards like GDPR and PCI-DSS, encryption in transit and at rest, and role-based access control (RBAC).

A testbed of artificial and actual transactional data streams is used to validate the methodology. Various fault injection scenarios, such as Kafka broker failures, delayed model responses, node crashes, and malformed data payloads, are presented. Mean Time to Detect (MTTD), Mean Time to Recover (MTTR), alert precision, system throughput, and prediction accuracy retention are used to assess the pipeline's performance. A thorough assessment of the advantages of the suggested architecture is made possible by establishing comparative baselines using non-resilient pipelines devoid of self-healing capabilities.

The pipeline's layered and modular architecture allows for near-zero downtime, continuous learning, and autonomous fault tolerance. This methodology allows for scalable and adaptive fraud detection. It establishes the architecture as a replicable pattern for robust AI deployment in compliance-sensitive sectors like digital payments, banking, and insurance.

4. Results

A series of controlled fault injection tests were conducted under production-like conditions to assess the efficacy of the suggested self-healing fraud detection pipeline using partially anonymized real-world datasets and synthetic financial transaction data. The study compared the system's detection and recovery capabilities to a baseline non-resilient pipeline devoid of autonomous recovery mechanisms. The following essential metrics were noted: throughput under stress, fraud detection accuracy, retention, Mean Time to Detect (MTTD), and Mean Time to Recover (MTTR).

The first set of experiments examined the failure scenarios microservice crashes, Kafka broker disruptions, model drift, and ingestion of corrupted data payloads typical in cloud-native environments. The Self-Healing Pipeline system uses Kubernetes control loops, alerting systems, and Prometheus-based telemetry to detect problems and start repair processes instantly. On the other hand, manual identification and intervention were necessary for the Non-Resilient Pipeline.

Figure 1 below contrasts the MTTD values for the two pipeline types. In every tested scenario, the self-healing architecture continuously achieved detection times under 15 seconds, with the microservice crash resulting in the lowest time of 5 seconds. Due to the lack of integrated observability, the non-resilient system, on the other hand, showed noticeably longer detection windows, averaging between 30 and 50 seconds.

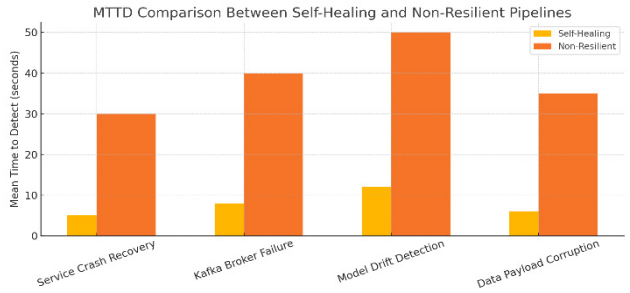

Figure 1: MTTD comparison of non-resilient and self-healing pipelines.

To determine how quickly the system could recover and start up fraud detection operations again after detection, MTTR was measured. Because of automated restart logic, model reloading, and traffic rerouting, the self-healing pipeline could restore services and model performance in less than 20 seconds in most scenarios, as seen in **Figure 2**. Due to its lack of automated retraining and deployment capabilities, the non-resilient pipeline, on the other hand, had MTTRs of more than 90 seconds and needed more than two minutes to stabilize in the worst-case scenario (model drift).

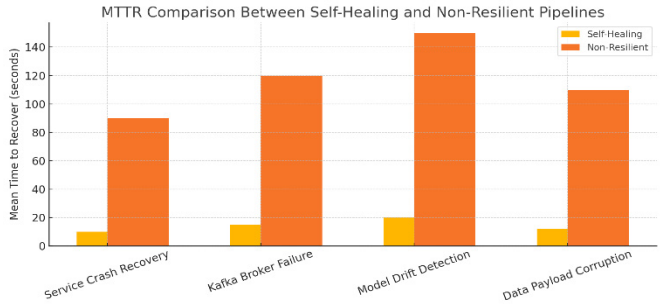

Figure 2: MTTR comparison of non-resilient and self-healing pipelines.

During most recovery operations, the self-healing pipeline kept model performance within an accuracy window of ± 2% in fraud detection accuracy. Precision and recall held steady even when concept drift caused dynamic retraining of the models. This was ascribed to fallback strategies and pre-configured validation metrics that triggered a previously validated model upon detecting anomalies in the current inference behavior.

Furthermore, under stress, system throughput, the quantity of transactions processed per second, exhibited excellent consistency. The Kafka-based architecture prevented data loss by guaranteeing buffering and replay capabilities during failure scenarios. During service healing operations, peak throughput slightly decreased (5–10%) but quickly returned to normal, suggesting graceful degradation instead of systemic failure.

These findings show that self-healing capabilities in cloud-native fraud analytics pipelines significantly increase fault resilience, decrease operational downtime, and maintain detection performance. Enterprise-grade deployments in financial institutions and digital commerce ecosystems are especially well-suited for the system's capacity to address errors and preserve compliance-critical functions proactively.

5. Discussion

The experimental assessment of the self-healing fraud detection pipeline strongly advocates implementing autonomous, cloud-native architectures in financial analytics settings. The findings highlight the vital benefits of incorporating automated remediation, fault tolerance, and observability into each stage of the fraud detection pipeline, from data ingestion to inference serving. In real-world applications, these features directly improve system availability, reduce operational costs, and maintain the effectiveness of fraud detection in the event of a failure.

One of the evaluation's most important findings was the significant decrease in Mean Time to Detect (MTTD) and Mean Time to Recover (MTTR) compared to non-resilient baselines. Kubernetes-native primitives (like liveness/readiness probes, horizontal pod autoscalers, and custom operators) in conjunction with Prometheus and Grafana-powered real-time metrics pipelines are primarily responsible for the responsiveness of the underlying architecture. Because of this integration, the system can instantly detect anomalies from latency spikes to prediction anomalies—and start remedial processes. On the other hand, traditional systems depend on human operators or static health checks, increasing the chance of fraud going undetected during these windows and lengthening detection and recovery times.

The capacity to manage model drift and data schema evolution without human intervention is a crucial component of machine learning pipeline resilience that is frequently disregarded in traditional fraud detection systems. Our system's integration with automated retraining pipelines, rollback tactics, and drift detection metrics guarantees that models remain performant and contextually relevant. This ability is essential when adversaries constantly change tactics, such as in financial fraud detection. The system can maintain high detection precision even when underlying patterns change because it can recognize and counteract these shifts independently.

Furthermore, fault isolation and graceful degradation were made possible by using Apache Kafka for asynchronous,

decoupled communication between services. Upstream and downstream components were not blocked by services that experienced delays or crashes, enabling the pipeline to continue processing unaffected data segments. The principles of microservices are well-aligned with this decoupling approach, which guarantees scalability, modularity, and testability while permitting fine-grained recovery actions. Preserving offset-based recovery and exact-once delivery semantics also promotes transactional integrity.

Another observation concerns the trade-off between observability overhead and performance. Adding sidecar containers, health probes, and monitoring tools introduces a quantifiable but tolerable resource overhead (~10-15% CPU and memory in stress tests). However, considering the substantial improvements in automated recovery, fault tolerance, and operational continuity, this expense is justified. This trade-off is functionally and financially feasible in production settings where compliance and uptime are crucial.

Through versioned model tracking, structured alert history, and centralized logging, the architecture naturally facilitates traceability from a regulatory and auditability perspective. This is particularly important for businesses that must adhere to financial compliance frameworks (like PCI-DSS, SOX, or GDPR), where it is mandatory to keep thorough records of system behavior and automated decision-making logic. The system ensures that every event is auditable, explicable, and compliant by recording all recovery actions, configuration modifications, and model redeployments.

However, several difficulties remain. First, careful policy engineering and continuous testing are necessary to create effective and generalizable remediation logic across failure types. Anomaly detection also carries the risk of false positives, which could result in instability by triggering needless remediation measures. Furthermore, the system's dependence on managed cloud services (for AI, storage, and computation) raises questions regarding cost control and vendor lock-in, especially as deployment scale increases.

Notwithstanding these drawbacks, the suggested architecture strongly aligns with the objectives of scalable system management, ongoing fraud surveillance, and intelligent automation. In terms of operational resilience, regulatory readiness, and technical sophistication, it is a significant improvement over legacy detection systems.

The conversation demonstrates that a well-designed self-healing pipeline is a workable, deployable tactic for contemporary fraud detection systems, not merely a theoretical ideal. Future fraud detection systems can be made scalable, flexible, and compliant by combining cloud-native technologies, real-time observability, and autonomous control loops.

6. Conclusion

Due to the growing complexity and speed of financial fraud in digital ecosystems, a new class of intelligent, robust, and scalable detection infrastructures is required. To maintain high fraud prediction accuracy and continuous processing capabilities, this paper has presented a cloud-native, self-healing fraud detection pipeline that autonomously handles application-level and infrastructure failures. The suggested system achieves continuous resilience and predictive reliability without requiring constant manual oversight by integrating microservices, Kubernetes-based orchestration, real-time observability tools, and feedback-driven remediation workflows.

Compared to conventional non-resilient pipelines, the experimental results show notable decreases in mean time to detect (MTTD) and mean time to recover (MTTR), confirming the architecture's efficacy. These gains were consistently seen in failure scenarios, such as ingesting malformed data, concept drift in ML models, Kafka broker disruptions, and service outages. While the automated control loops and model health monitoring allowed for the prompt detection and correction of prediction anomalies, the use of Kafka for event streaming offered resilience against service outages. These features position the suggested pipeline as a workable, ready-to-use solution for businesses operating in highly regulated, real-time financial sectors.

This study also emphasizes the broader significance of integrating MLOps workflows, AI-driven observability, and DevOps principles. The pipeline's flexibility in the face of hostile activity and changing fraud trends is improved by its ability to recognize model performance degradation and automatically retrain or rollback to earlier iterations. The system is also easily expandable and portable across various cloud environments due to its modular architecture and compliance with open standards, providing deployment flexibility while adhering to audit and security regulations.

Notwithstanding the advancements, there are still certain restrictions on the work. Effective remediation logic design for complex failure modes is still challenging and needs careful validation and tuning to prevent misdiagnosis or false recoveries. Furthermore, future systems might profit from incorporating more autonomous learning paradigms like reinforcement learning or federated adaptation to support increased self-awareness and contextual intelligence, even though the current solution supports deterministic retraining based on data and metric thresholds. Explainability will play an increasingly important role in automated decisions as systems become more complex, especially in sectors where regulatory agencies require openness regarding the decision-making process or the reasons behind a computerized recovery.

Future research will focus on enhancing root cause analysis with graph-based observability frameworks, developing model governance techniques for safe and moral AI deployments, and adding multi-cloud and edge-compatible versions to broaden the pipeline's capabilities. Furthermore, incorporating blockchain-based audit trails and decentralized identities may improve the pipeline's credibility in delicate settings.

This paper concludes by showing that self-healing cloud-native fraud detection pipelines are both strategically required and technically possible. They are crucial in creating intelligent, constantly available, and legally compliant infrastructures that maintain operational excellence while protecting against contemporary fraud threats. This work paves the way for autonomous security operations in the digital financial era by laying the groundwork for a new generation of self-sustaining and predictive fraud analytics systems.

9. References

1. S Bhattacharyya S Jha, K Tharakunnel, et al. Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 2011; 50: 602-613.

2. D Pozzolo, O Caelen, Y Le Borgne, et al. Credit card fraud detection: A realistic modeling and a novel learning strategy," *IEEE Trans. Neural Netw. Learn. Syst,* 2018; 29: 3784-3797.

3. A Dal Pozzolo, G Boracchi, O Caelen, et al. Credit card fraud detection: A realistic modeling and a novel learning strategy. In: *Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA)*, 2015; 1-10.

4. L Neumeyer, B Robbins, A Nair, et al. Apache Flink: Stream and Batch Processing in a Single Engine. *IEEE Data Eng. Bull,* 2015; 38: 28-38.

5. B Burns, B Grant, D Oppenheimer, et al. Borg, Omega, and Kubernetes. *Commun. ACM*, 2016; 59: 50-57.

6. JO Kephart, DM Chess. The vision of autonomic computing. *Computer*, 2003; 36: 41-50.

7. M Fowler. Microservices: A definition of this new architectural term. *Martinfowler.com*, 2014.

8. R Burns. Monitoring Cloud-Native Applications Using Prometheus and Grafana. *ACM Queue*, 2021;19: 23-34.

9. S Sculley. Hidden Technical Debt in Machine Learning Systems. In: *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2015; 2503-2511.

10. M Baier, R Kliegl, A Rehmsmeier. Detecting concept drift in classification problems. *Expert Syst. Appl,* 2020; 150: 113290.

11. V Gupta, A Vashishtha, P Kumar. Adaptive fraud detection using cloud-native microservices. In: *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, 2020; 207-214.

12. A Kundu, Sharma. Secure and Scalable Fraud Detection in Cloud Environments. In: *Proc. Int. Conf. Cyber Secur. Cryptogr. (ICCSC)*, 2021; 89-95.

13. R Castro Fernandez, M Migliavacca, E Kalyvianaki, et al. Integrating Scale-Out Stream Processing with Declarative Data Analytics. In: *Proc. ACM Symp. Cloud Comput. (SoCC)*, 2021; 219-233.