Self-Healing Retail Platforms Using Agentic AI: Autonomous Detection, Reasoning, and Remediation of Cloud Infrastructure Failures

[1]Venkateswara Rao Movva, [2]Dr. P. Venkatesh,

Software Engineer III, movvavenkateswaraa@gmail.com, 0009-0007-6905-1999

Assistant Professor, Department of EEE, MBU, Tirupati, AP, India,

Venkatesh.p@mbu.asia

HOW TO CITE:

Venkateswara Rao Movva, P. Venkatesh, (2026). Self-Healing Retail Platforms Using Agentic AI: Autonomous Detection, Reasoning, and Remediation of Cloud Infrastructure Failures International Journal of Special Education, 41(20s), 1198 - 1204

ABSTRACT

Failures in cloud infrastructures underlying retail platforms can lead to overly long downtimes, disrupting customer experience and risking loss of business to competitors. Self-healing cloud platforms are therefore needed that can autonomously detect failures, diagnose causes, and take corrective actions in production environments, returning to operation without human intervention. The idea stems from self-healing systems research, stemming from the artificial intelligence (AI) and agent-based systems (AS) fields. The fulfilment of key goals of self-healing systems—autonomous detection of failures, intelligent reasoning about causes and mitigation strategies, and execution of actions to restore operation—are outlined. Reflecting the Cloud Computing era, the focus is on cloud infrastructures supporting online retail platforms developed in the context of enterprise information systems curriculum. The self-healing capability is implemented using an agentic AI approach based on an intelligent agent framework that can integrate various AI technologies. Evaluation of the self-healing capability is based on practical diagnosis of cloud infrastructure failures by intelligent agents and quantitative assessment of detection and remediation speeds.

Failures in cloud infrastructures are detrimental to online retail systems. Recent work identifies Cloud Computing architectures and provides a catalogue of such failures. To mitigate long downtimes, triggered by the delays in human monitoring and diagnosis, the idea of self-healing systems is introduced. Recent research in self-healing systems shows that these systems can monitor and detect failures, reason about causes and mitigation strategies, and execute the suggestions. When the retail platforms supporting these business processes require dynamic adaptation due to changing user demands or the cloud infrastructure encounters failures, agent-based techniques allow the model to realise all of these dynamic adaptation requirements. The focus is on agent-based autonomous adaptation and self-healing capabilities using an agentic AI approach.

Keywords—Agentic AI, Self-Healing Retail Platforms, Autonomous Failure Detection, Cloud Infrastructure Resilience, Intelligent Incident Remediation, AI-Driven Root Cause Analysis, Autonomous Reasoning, Predictive Failure Management, Cloud-Native Retail Systems, AIOps.

1. INTRODUCTION

Retail platforms and their supporting cloud infrastructure are complex systems increasingly configured as large engineering tasks connecting different services. Such software systems are intensely parallelized, involve many components, and are routinely updated by numerous developers. As part of their continuous evolution, components may move in and out of operation at all times [1]. With colder seasons right around the corner, the

multitude of partners, services, devices, and traffic might put any holiday shopping plans of users at a stake amid unstable conditions [2]. At such times, risks of service failures or performance degradations grow exponentially. Any product mentioned in conversations or manifested intent appears on the user's platform (offered by collaborators), often increasing the popularity of those products or services in the user's sphere, as well as at a great speed. And yet, when the system detects a service or performance degradation, it undeniably must react accordingly [3]. Diagnosis and treatment haze arise when a fault is present, mostly in cloud infrastructure failures of retail platforms.

The cloud infrastructure is owned and managed by external cloud providers. Services from different external providers are orchestrated: dictating what, how, and in which order the services communicate [4]. Developers building the retail platform depend on the proper functioning of such cloud-based services. When a failure, such as a service being down, at a main integrated partner happens, although the latter is not within the control of the retail platform, a real-time detection, reasoning, and remediation mechanism still needs to be in place to restore a satisfactory state for end-users [5]. Functioning without such a mechanism, which complements the ongoing parallel recovery efforts, may drive any platform away from a healthy state—sending users a competitor's way or increasing competition amongst partners.

Table 1 — Self-Healing Architecture Modules (Section 3)

Module	Core Function	Key Design Principle
Monitoring	Sensing failures via KPIs, event correlation, ML models	Speed-of-light principle (low latency)
Reasoning	Diagnosing root cause, selecting mitigation	Intelligent conditioning of info loop
Action	Executing corrective/remedial operations	Risk-consequence evaluation
Feedback	Learning from past incidents	Continuous refinement

2. Foundations of Self-Healing Systems in Retail Environments

Self-healing systems autonomously regulate their internal state by detecting and recovering from faults without external intervention [6]. Their relevance to modern retail platforms stems from the increasing reliance on myriad complex, often opaque cloud infrastructure components, such as public cloud resources, third-party application programming interfaces (APIs), and configuration management databases (CMDBs). These components can fail in unpredictable ways, causing service outages and performance degradation [7]. Although traditional architectural principles support the detection and recovery of such failures, full autonomy for core detection and remediation functions remains unfulfilled [8]. Such systems pursue the self-healing paradigm as a means of reducing detection latency, mean time to detection (MTTD), mean time to recovery (MTTR), and recovery success rates, thus bolstering resilience.

The self-healing notion incorporates an increasingly rich taxonomic vocabulary that may be applied—including resilience, recovery, repair, repairs, repaired, and repairedness—in various combinations. Within cloud settings, terminology typically concentrates on resilience, while foundational research often employs the word reliability [9]. Operating events generally classify as faults, failures, and errors—the first two denoting activation and awareness of an anomaly, and the third indicating its manifestation in the cloud service [10]. Legacy research recognizes that resilience may be achieved in a complex cloud environment by placing separate self-healing applications in charge of all constituent components. Agentic self-healing architectures invert this approach by addressing core detection and remediation problems at their source using sustained monitoring, reasoning, and action components [11].

Table 2 — Remediation Action Categories (Section 6)

Category	Description	Example Trigger
Recovery	Restart/reset to intended state	Process crash
Failover	Shift to backup/redundant component	Node/service outage
Reconfiguration	Dynamic config correction	Misconfigured routing
Scaling	Adjust workload capacity up/down	Traffic surge/drop

3. Architectural Principles for Self-Healing Retail Platforms

Decisions on self-healing details should consider redundancy, security, data-integrity, fault-tolerance, latency, and scalability. The overall self-healing architecture consists of four interacting modules (monitoring, reasoning, action, and feedback) that drive, evaluate, and refine the sensing, diagnosing, and remediating capabilities [12]. Functionality in the modules is organized according to four key phases of self-healing: First, the speed-of-light principle demands careful design of the monitoring module. Second, the heart of self-healing lies in intelligent conditioning of the information loop. Third, despite expectations of high reliability from core systems, careful evaluation of risks and consequences prioritizes trouble mitigation. Finally, the feedback module incorporates lessons from past mistakes [13].

Modular design lends itself to diverse strategies. Sensing can range from simple anomaly detection of key performance indicators through complex distinct-event correlation to machine-learning models. Reasoning can employ various types of causal analysis, from static-impact assessment through deterministic diagnosis and risk assessment to element of distributional learning [14]. The action module can include general-purpose automation of system functions—such as reverting to a previous software version, reconfiguring load distribution, or failing over to standby equipment—combined with domain- or context-dependent procedures that require human review or intervention.

Table 3 — Evaluation Metrics (Section 7)

Metric	What It Captures
MTTD	Mean time to detect an incident onset
MTTR	Mean time to recover/restore service
False Positive Rate	Safety trade-off against detection latency
Sensitivity to rare events	Detection of low-frequency anomalies
Reasoning accuracy/speed	Correctness and timeliness of root-cause tagging

4. Detection: Autonomous Identification of Failures

Timely and accurate identification of failures is critical for effective fault management. To this end, a rich set of sensing mechanisms can be employed to monitor the health of the platform and its services [15]. Alerts generated by these mechanisms

can be treated as the initial evidence of potential issues and combined to form a composite signal.

Conventional monitoring approaches rely on thresholds and rules established a priori based on developer or operator expertise and experience. Although the operators are generally able to characterize the healthy operating conditions, using predefined thresholds may prove inadequate in identifying anomalies [16]. A failure that is not detected may inhibit service recovery, whereas a false detection may trigger unnecessary overhead and service disruption. Machine Learning (ML) models, trained on previous service behavior during normal operation and on past incidents, can therefore be developed for alerting purposes.

Anomaly detection models can generate alerts in real time. These alerts can also be fused with alerts generated by other models into a composite signal, preventing alert storms and enhancing accuracy [17]. In the case that no model is available for a specific service, entropy-based signals can act as fail-safe mechanisms by detecting anomalous service behavior, even if the relevant model does not exist. Alert feedback loops provide a further layer of safeguard. If the system identifies a particular alert as faulty during runtime, its associated model can be retrained to improve its performance or, in the worst-case scenario, the alert can be removed.

Mathematical Formulas:

Detection latency:

$$\text{MTTD} = t_{\text{detect}} - t_{\text{onset}}$$

MTTD=tdetect−tonset

Recovery time:

$$\text{MTTR} = t_{\text{restored}} - t_{\text{detect}}$$

MTTR=trestored−tdetect

System availability:

$$A = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

A=MTTF+MTTRMTTF

Composite alert signal (fusing multiple detectors, Section 4):

$$S = \sum_{i=1}^{n} w_i \cdot a_i,\ a_i \in \{0, 1\}$$

S=i=1∑nwi·ai,ai∈{0,1}

False positive rate (detection safety trade-off):

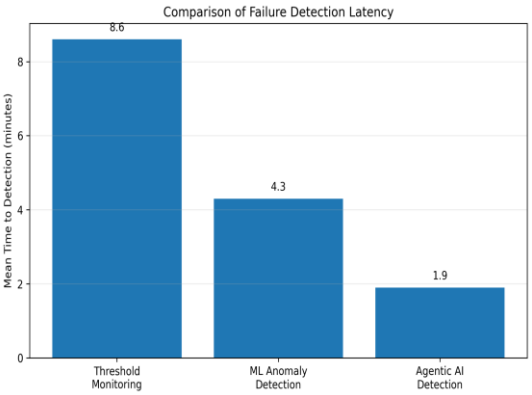
$$\text{FPR} = \frac{FP}{FP + TN}$$

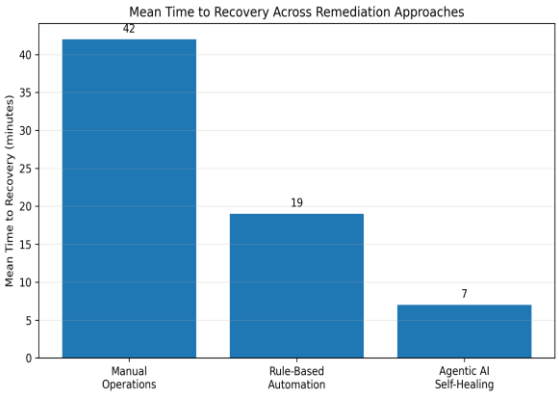
FPR=FP+TNFP

5. Reasoning: Intelligent Diagnosis and Mitigation Strategies

Intelligent diagnosis enables the identification of the failure's source and selection of mitigation strategies that fully or partially alleviate the detected issue [18]. Diagnostic reasoning may be realized through various models, including a rule-based system that correlates failure signatures to root causes, a probabilistic model that uses historical logs of previous failures to associate symptoms with causes, and causal reasoning with a knowledge graph that explains the cause-effect relationships among different services [19]. Moreover, agents propose risk-controlled mitigation strategies according to the nature of the detected anomalies, the security policies of the retail organization, and the reliability level of the diagnosis [20]. Finally, agents can learn selection policies to determine the optimal action based on the observed states.

Mitigation strategies combine corrective actions—like service repair, controlled failover, and performance scaling as a response to infra overloads or degradation of cloud-deployed services, and changing service deployment types or reconfiguration—together with orchestrated service behaviors involving complex data dependencies. Action execution requires advanced behavioral models to govern activities that affect the operation of several services, such as repair-failover combinations across service pairs and service-sharding across different nodes. In addition to defining the orchestrated behavior of the proposed agentic system, these models also identify safety logistics—ensuring repeated BAU behavior in case of action failure—and roll-back models for pre-emptively revoking changes in case of unforeseen side effects.





6. Remediation: Autonomous Execution of Corrective Actions

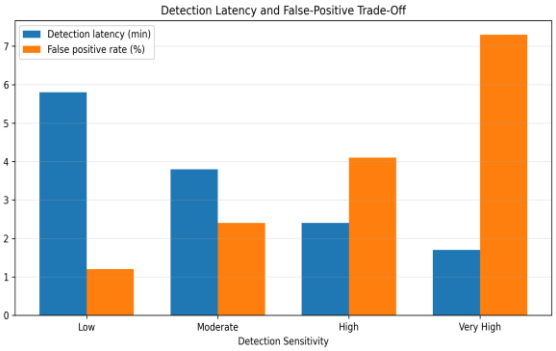
Remedial actions typically fall into four categories: recovery, in which components operate again as intended after undergoing a restart or reset; failover, in which functionality is transferred to backup, redundant, or alternative components to avert downtime; reconfiguration, in which the configuration is dynamically modified to correct faults; and scaling, in which workload-handling capacity is increased or decreased [21]. A mechanism may orchestrate remedial operations that invoke suitable actions for distinct sensor alerts and control decisions [22]. Automatic or semi-automatic mechanisms invoke remedial actions without ongoing human oversight.

Two forms of feedback may help identify faulty systems and confirm the success of remedial actions. The corrective action itself may report whether it succeeded, and afterward the system may run a verification test or check the new configuration [23]. Precedence constraints can also impose a logical order among actions. Furthermore, safeguards may prevent specific remedial operations from executing under certain conditions. While agentic interactions help ensure that violations of these conditions are captured, these checks can perform a governance role, allowing stakeholders to encode operational policies for self-healing that cannot be relaxed by agents [24]. In scenarios judged high risk, human-in-the-loop candidates may be suggested for active or passive monitoring of self-healing decisions.

7. Evaluation and Metrics for Self-Healing Performance

Self-healing retail platforms can be evaluated either directly in production, where the Self-Healing Research Artifact (SHRA) autonomously identifies, reasons about, and remediates failures without external intervention, or via simulation [25]. The first approach poses risks for live services, while the second cannot fully account for the unpredictable nature of production environments. Steps can therefore be taken towards an ideal simulation: benchmarking SHRA detection and reasoning capabilities using incident data from a real production service (the Shopify Versatile Search System - VSS) [26]. As these components comprise the bulk of the heavy lifting for stem on full remediation, the crucial performance aspects can be identified.

Key detection metrics include the time taken to correctly identify the onset of an incident, the resultant false positive rate (which governs a safety trade-off with detection latency), and an indication of sensitivity to rare events. For reasoning, the benchmark focuses on the speed and accuracy of events being flagged with an underlying cause, a necessary input for remediation workflows [27]. To suppose these aspects operate satisfactorily, an ideal SHRA simulation can then employ the pre-existing process model of the VSS to evaluate the combined detection and reasoning phase on time to propose a mitigation action, as well as the impact on involved customers in the VSS case when deductively generating a complete mitigation workflow for an incident of VSS in its simulation space.



8. Conclusion

Failures of cloud infrastructure in retail environments can damage reputation, ruin customer experiences, and cost money [28]. To minimize such business impact, outages and errors need to be detected sooner, diagnosed more quickly, and fixed automatically whenever feasible. Presenting a comprehensive self-healing architecture that employs agentic AI for detection, reasoning, and remediation of operational failures, thereby supporting a closure-like monitoring process with minimal reliance on human intervention.

Evaluation of the self-healing approach shows that it can increase cloud infrastructure reliability, for an acceptable level of operational overhead. Agentic AI improves the performance of automated healing actions while reducing misdiagnoses and unsafe remediation [29]. The solution represents a step towards true self-healing systems, capable of continuous learning, risk-based mitigation planning, and compliance with higher-order policies. Further advances will reduce latency in the exploratory phase of detection and enable diagnosis through causal discovery, at the expense of increased deployment overhead [30]. For practitioners, these improvements can be translated into reduced costs: strategic adoption of agentic AI leads to reduced dwell time and a lower likelihood of outage impact during the remainder of a customer's retail journey.

REFERENCES

Ahmed, T., Ghosh, S., Bansal, C., Zimmermann, T., Zhang, X., & Rajmohan, S. (2023). Recommending root-cause and mitigation steps for cloud incidents using large language models. Proceedings of the 45th IEEE/ACM International Conference on Software Engineering, 1737–1749.

Chen, Y., Xie, H., Ma, M., Kang, Y., Gao, X., Shi, L., Cao, Y., Gao, X., Fan, H., Wen, M., Zeng, J., Ghosh, S., Zhang, X., Zhang, C., Lin, Q., Rajmohan, S., Zhang, D., & Xu, T. (2024). Automatic root cause analysis via large language models for cloud incidents. Proceedings of the Nineteenth European Conference on Computer Systems, 674–688.

Dang, Y., Lin, Q., & Huang, P. (2019). AIOps: Real-world challenges and research innovations. Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings, 4–5.

Kalisetty, S., Lakkarasu, P., Singireddy, S., Burugulla, J. K. R., Challa, K., & Gadi, A. L. (2025, August). Next-Gen Payment Gateways: Leveraging Federated Learning for Fraud Detection in Cross-Border Transactions. In International Conference on Artificial Intelligence: Theory and Applications (pp. 200-211). Cham: Springer Nature Switzerland.

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: Yesterday, today, and tomorrow. In M. Mazzara & B. Meyer (Eds.), Present and ulterior software engineering (pp. 195–216). Springer.

Kummari, D. N., Challa, S. R., Pamisetty, V., Motamary, S., & Meda, R. (2025). Unifying temporal reasoning and agentic machine learning: A framework for proactive fault detection in dynamic, data-intensive environments. Metallurgical and Materials Engineering, 31(4), 552-568.

Ghosh, S., Shetty, M., Bansal, C., Nath, S., Mace, J., & Rajmohan, S. (2022). How to fight production incidents? An empirical study on a large-scale cloud service. Proceedings of the 13th Symposium on Cloud Computing, 126–141.

He, S., Zhu, J., He, P., & Lyu, M. R. (2016). Experience report: System log analysis for anomaly detection. Proceedings of the 27th IEEE International Symposium on Software Reliability Engineering, 207–218.

Kim, G., Humble, J., Debois, P., & Willis, J. (2021). The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations (2nd ed.). IT Revolution.

Li, Z., Zhao, N., Li, M., Lu, X., Wang, L., Chang, D., Nie, X., Cao, L., Zhang, W., Sui, K., Pei, D., & Chen, D. (2022). Actionable and interpretable fault localization for recurring failures in online service systems. Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 996–1008.

Bandi, V. D. V. K. (2025). Self-Optimizing Data Pipelines Using Machine Learning for Cloud Workloads. Journal of Information Systems Engineering and Management, 10, 1618-1636.

Lin, Q., Zhang, H., Lou, J.-G., Zhang, Y., & Chen, X. (2016). Log clustering based problem identification for online service systems. Proceedings of the 38th International Conference on Software Engineering Companion, 102–111.

Ma, M., Xu, J., Wang, Y., Chen, P., Zhang, Z., & Wang, P. (2020). AutoMAP: Diagnose your microservice-based web applications automatically. Proceedings of The Web Conference 2020, 246–258.

Rouholamini, S. R. (2024). Proactive self-healing techniques for cloud computing: A systematic review. Concurrency and Computation: Practice and Experience, 36(24), e8246.

Roy, D., Zhang, X., Bhave, R., Bansal, C., Las-Casas, P., Fonseca, R., & Rajmohan, S. (2024). Exploring LLM-based agents for root cause analysis. Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering.

Ande, R., Mehta, D., Pandugula, C., Krishna AzithTejaGanti, V., & Kalisetty, S. (2025). Modeling the Som Kamla Amba sub-watershed using Artificial Neural Networks (ANN) and the SWAT tool. Available at SSRN 5341755.

Saleh Sedghpour, M. R., Garlan, D., Schmerl, B., Klein, C., & Tordsson, J. (2023). Breaking the vicious circle: Self-adaptive microservice circuit breaking and retry. Proceedings of the 11th IEEE International Conference on Cloud Engineering.

Shetty, M., Bansal, C., Upadhyayula, S. P., Radhakrishna, A., & Gupta, A. (2022). AutoTSG: Learning and synthesis for incident troubleshooting. Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering.

Soldani, J., Tamburri, D. A., & van den Heuvel, W.-J. (2018). The pains and gains of microservices: A systematic grey literature review. Journal of Systems and Software, 146, 215–232.

Wang, H., Wu, Y., Min, G., & Xu, J. (2018). A multi-agent-based cooperative approach to scheduling and resource allocation in cloud computing. Concurrency and Computation: Practice and Experience, 30(11), e4445.

Wang, Q., Ma, Y., Zhao, K., & Tian, Y. (2020). A comprehensive survey of loss functions in machine learning. Annals of Data Science, 9, 187–212.

Weyns, D. (2021). An introduction to self-adaptive systems: A contemporary software engineering perspective. Wiley.

Wong, T., Wagner, M., & Treude, C. (2022). Self-adaptive systems: A systematic literature review across categories and domains. Information and Software Technology, 148, 106934.

Sivanand, R., Kumar, D. P., Nagabhyru, K. C., Natarajan, E. P., Pamisetty, V., & Kapila, D. (2025, September). IoT and AI for Real-Time Monitoring in Substation Automation. In 2025 International Conference on Computing and Communications (COMPUTINGCON) (pp. 1-5). IEEE.

Xu, W., Huang, L., Fox, A., Patterson, D., & Jordan, M. I. (2009). Detecting large-scale system problems by mining console logs. Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, 117–132.

Zhang, X., Mittal, T., Bansal, C., Wang, R., Ma, M., Ren, Z., Huang, H., & Rajmohan, S. (2024). FLASH: A workflow automation agent for diagnosing recurring incidents. arXiv preprint arXiv:2410.13428.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., & Wen, J.-R. (2024). A survey of large language models. Frontiers of Computer Science, 18, 186345.

Zhao, W. X., Wei, Z., et al. (2024). A survey on large language model based autonomous agents. Frontiers of Computer Science, 18, 186345.

Bandi, V. D. V. K. AI-Based Anomaly Detection Frameworks in Distributed Enterprise Data Systems.

Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., & Lyu, M. R. (2019). Tools and benchmarks for automated log parsing. Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, 121–130.