International Journal of Emerging Trends in Engineering and Management Research (IJETEMR)

|ISSN: 2455-7773 | www.ijetemr.com| A Double-Blind Peer-Reviewed and Bimonthly Scholarly Journal|

|Volume 11, Issue 4, July-August 2026|

DOI:10.15662/ijetemr.2026.1104002

Self-Healing AI Automation Frameworks for Highly Resilient Cloud-Native Software Delivery Pipeline Management

Praiselin Lydia Gladston

Rogers Communications, Calgary, Alberta, Canada

Publication History: Received: 05.06.2026; Revised: 02.07.2026; Accepted: 07.07.2026; Published: 20.07.2026

ABSTRACT: Cloud-native software delivery has transformed application development through microservices, containers, Kubernetes orchestration, infrastructure as code, continuous integration, and continuous delivery. However, the increasing complexity and dynamism of cloud environments also create new reliability challenges, including configuration drift, resource exhaustion, deployment failures, service dependency problems, security incidents, and infrastructure instability. Conventional DevOps automation primarily executes predefined procedures and often depends on human intervention when unexpected conditions occur. Self-healing artificial intelligence (AI) automation frameworks offer an emerging approach in which intelligent systems continuously observe delivery pipelines, identify abnormal conditions, diagnose probable causes, predict failures, and automatically execute corrective actions. This study examines the conceptual design and operational characteristics of self-healing AI automation frameworks for resilient cloud-native software delivery pipeline management. The proposed research methodology integrates systematic literature analysis, framework modelling, scenario-based experimentation, and quantitative evaluation using resilience-oriented metrics. The study focuses on integrating observability, machine learning, anomaly detection, causal diagnosis, automated remediation, policy enforcement, and continuous feedback into a closed-loop delivery architecture. Particular attention is given to balancing autonomous remediation with safety controls, explainability, security, and human oversight. The research argues that self-healing pipelines can reduce recovery time, improve deployment reliability, minimize operational intervention, and strengthen service continuity when designed with appropriate guardrails. The resulting framework provides a foundation for developing adaptive, resilient, and intelligent cloud-native software delivery ecosystems.

KEYWORDS: self-healing AI, cloud-native computing, DevOps automation, software delivery pipelines, artificial intelligence, machine learning, Kubernetes, resilience engineering, automated remediation, observability, continuous delivery, anomaly detection

I. INTRODUCTION

Cloud-native software engineering has become a dominant approach for delivering scalable digital services because it enables organizations to combine microservices, containers, orchestration platforms, infrastructure as code, continuous integration, continuous delivery, and elastic cloud resources. Technologies such as Kubernetes have further encouraged the development of distributed systems capable of dynamically scaling and recovering from certain infrastructure failures. Nevertheless, cloud-native architectures introduce substantial operational complexity. A single application may depend on hundreds of services, container instances, databases, message brokers, application programming interfaces, network components, and cloud resources. Changes to any component can influence the behaviour of the wider system. Consequently, maintaining reliability throughout the software delivery lifecycle has become increasingly difficult.

Traditional CI/CD pipelines automate activities such as source-code compilation, testing, security scanning, container image construction, deployment, and rollback. Although these capabilities significantly reduce manual work, most conventional pipelines remain fundamentally rule-driven. They execute predetermined responses to recognized events but may fail when they encounter novel combinations of failures. For example, a deployment may technically succeed while creating abnormal latency because of an interaction between a newly introduced service, resource limits, network congestion, and a downstream dependency. Identifying and resolving such incidents can require engineers to interpret large quantities of logs, metrics, traces, deployment information, and configuration data.

|ISSN: 2455-7773 | www.ijetemr.com| A Double-Blind Peer-Reviewed and Bimonthly Scholarly Journal|

|Volume 11, Issue 4, July-August 2026|

DOI:10.15662/ijetemr.2026.1104002

Artificial intelligence provides an opportunity to extend pipeline automation from predetermined execution toward adaptive decision-making. Machine-learning models can identify abnormal behaviour, predict potential failures, correlate events, classify incidents, and recommend or execute remediation actions. When these capabilities are integrated into a closed feedback loop, a delivery platform can potentially detect operational problems, diagnose their causes, apply corrective measures, and verify whether recovery has occurred. Such systems are commonly described as self-healing systems because they seek to restore desired operational states with limited human intervention.

Self-healing automation is particularly relevant to cloud-native delivery because the environment is highly dynamic. Workloads can be created and destroyed automatically, infrastructure can scale rapidly, deployments can occur several times a day, and dependencies can change independently. A resilient pipeline therefore needs more than fast deployment; it needs continuous awareness of system health and the ability to respond safely to changing conditions. The central research problem is consequently how AI-based automation can be incorporated into software delivery pipelines without introducing unacceptable risks associated with incorrect predictions, unsafe remediation, opaque decisions, or excessive automation.

This study investigates a self-healing AI automation framework designed for highly resilient cloud-native software delivery pipeline management. It considers the integration of observability, intelligent anomaly detection, diagnosis, predictive analytics, automated remediation, policy-based controls, and feedback mechanisms. The research also examines how resilience can be evaluated through measurable indicators such as mean time to detection, mean time to recovery, deployment failure rate, recovery success rate, availability, change failure rate, and human intervention frequency. The objective is to establish a structured framework through which AI can complement DevOps practices while maintaining reliability, security, explainability, and appropriate human oversight.

II. LITERATURE REVIEW

The literature on cloud-native software delivery, DevOps, resilience engineering, AIOps, autonomous computing, and self-healing systems provides the conceptual foundation for this research. DevOps established the principle that development and operations should be integrated through automation, collaboration, continuous feedback, and rapid delivery. Continuous integration and continuous delivery subsequently enabled organizations to automate increasingly large portions of the software release lifecycle. However, continuous delivery itself does not guarantee resilience. A pipeline can deliver software rapidly while still producing frequent failures if testing, observability, configuration management, or recovery mechanisms are inadequate.

Research on autonomous and self-healing computing provides an earlier conceptual basis for intelligent operational automation. IBM's autonomic computing vision proposed systems capable of self-configuration, self-optimization, self-healing, and self-protection. The underlying principle was that complex computing environments should manage themselves according to defined objectives rather than requiring constant human administration. Modern cloud-native systems provide an especially suitable environment for realizing these ideas because orchestration platforms already possess mechanisms for restarting failed workloads, rescheduling containers, scaling resources, and maintaining desired states.

The Kubernetes ecosystem demonstrates an important distinction between conventional orchestration-based recovery and AI-driven self-healing. Kubernetes controllers continuously compare actual system state with a desired state and initiate corrective actions when discrepancies occur. This provides a foundational feedback-control mechanism. However, traditional controllers generally operate according to explicit rules rather than learning from historical operational data. AI can potentially extend this model by recognizing subtle behavioural patterns, predicting failures before they become service-impacting events, and selecting among alternative remediation strategies.

AIOps research has increasingly explored artificial intelligence for IT operations. AIOps platforms aggregate metrics, logs, traces, events, and other operational information to identify anomalies and reduce alert noise. Machine-learning approaches can establish baselines for normal behaviour and identify deviations that may indicate incidents. Event correlation can also associate multiple symptoms with a common underlying problem. These capabilities are important for self-healing pipelines because automated remediation depends on reliable detection and diagnosis. Nevertheless, anomaly detection alone does not constitute self-healing. An effective autonomous system must connect detection with decision-making, action execution, and verification.

|ISSN: 2455-7773 | www.ijetemr.com| A Double-Blind Peer-Reviewed and Bimonthly Scholarly Journal|

|Volume 11, Issue 4, July-August 2026|

DOI:10.15662/ijetemr.2026.1104002

Observability has therefore become an essential component of resilient cloud-native architectures. The widespread adoption of distributed tracing, structured logging, application metrics, and OpenTelemetry-based instrumentation provides richer evidence about system behaviour. Observability enables AI models to examine relationships among services and identify patterns that cannot be inferred from a single metric. For example, an increase in request latency combined with a specific deployment, memory pressure, and errors in a downstream service may provide stronger evidence of a causal problem than any individual signal.

Another important area of literature concerns chaos engineering and resilience testing. Chaos engineering deliberately introduces controlled failures to determine whether systems can withstand unexpected conditions. Practices associated with platforms such as Netflix's Chaos Monkey demonstrated the value of testing distributed systems against infrastructure failures rather than assuming that theoretical redundancy guarantees resilience. Integrating chaos experiments with AI-based automation could provide valuable training and validation data for self-healing systems. AI agents could learn which remediation strategies are effective under specific failure conditions and subsequently apply those strategies during production incidents.

Machine learning has also been investigated for failure prediction and root-cause analysis. Supervised learning can classify known incident types, while unsupervised and semi-supervised techniques can detect previously unseen anomalies. Time-series models can forecast resource exhaustion, latency increases, or capacity constraints. Graph-based techniques can represent service dependencies and help identify probable propagation paths. More recently, large language models have created possibilities for reasoning over heterogeneous operational information, generating remediation plans, interpreting logs, and assisting engineers. However, autonomous use of generative AI introduces significant concerns because language models can produce plausible but incorrect recommendations. Consequently, AI-generated actions should be constrained by policies, validated against system state, and executed through controlled interfaces.

Research on DevSecOps further demonstrates that resilience cannot be separated from security. An automated system with permission to modify infrastructure, deployments, or configurations becomes a high-value operational component and potentially an attractive target for attackers. Self-healing mechanisms must therefore include authentication, authorization, auditability, least-privilege access, policy enforcement, and rollback mechanisms. Automated remediation that restores availability while weakening security controls would represent an undesirable trade-off. Security must consequently be treated as a constraint on autonomous action rather than as a separate post-deployment activity.

Despite these advances, a gap remains between individual technologies and a comprehensive self-healing AI framework for the entire cloud-native software delivery pipeline. Existing research frequently concentrates on particular functions such as anomaly detection, predictive maintenance, Kubernetes recovery, AIOps, or automated incident response. Less attention is given to integrating these capabilities into a unified delivery lifecycle in which intelligence operates across code changes, testing, deployment, runtime monitoring, diagnosis, remediation, and post-recovery learning. This research addresses that gap by conceptualizing self-healing as an end-to-end closed-loop capability rather than a single automated recovery mechanism.

III. RESEARCH METHODOLOGY

The proposed research adopts a design-oriented and empirical methodology to investigate how self-healing AI automation can improve resilience in cloud-native software delivery. The methodology combines a structured literature review, requirements analysis, conceptual framework development, prototype-oriented modelling, controlled experimentation, and quantitative evaluation. The research begins by defining the problem domain and identifying the principal failure conditions affecting cloud-native delivery pipelines. These include build failures, test failures, deployment errors, configuration inconsistencies, container crashes, resource exhaustion, service degradation, dependency failures, network faults, security-policy violations, and cascading incidents. The purpose of this classification is to establish a representative failure taxonomy against which the framework can be evaluated.

The first methodological stage consists of a systematic analysis of existing academic and technical literature concerning self-healing computing, AIOps, DevOps, cloud-native engineering, machine learning for operations, resilience engineering, chaos engineering, observability, automated remediation, and intelligent incident management. Relevant

|ISSN: 2455-7773 | www.ijetemr.com| A Double-Blind Peer-Reviewed and Bimonthly Scholarly Journal|

|Volume 11, Issue 4, July-August 2026|

DOI:10.15662/ijetemr.2026.1104002

literature is organized according to four dimensions: detection, diagnosis, decision-making, and recovery. The review identifies technologies, algorithms, architectures, evaluation measures, and limitations reported by previous researchers. It also distinguishes between rule-based automation and adaptive AI-based automation. The resulting synthesis is used to identify functional requirements for the proposed framework.

The second stage develops a conceptual architecture based on a closed-loop control model. The framework contains six principal logical layers: software delivery, observability, intelligence, decision and policy control, remediation, and learning. The software-delivery layer manages source-code integration, automated testing, artifact creation, security validation, infrastructure provisioning, deployment, and rollback. The observability layer collects metrics, logs, traces, events, configuration information, deployment metadata, and service-level indicators. The intelligence layer applies statistical methods and machine-learning models to detect anomalies, predict failures, correlate events, and estimate root causes.

The decision and policy layer represents a critical safety boundary. AI should not have unrestricted authority to perform arbitrary operations. Instead, possible actions are evaluated according to predefined policies, operational risk, confidence levels, service criticality, and authorization requirements. Low-risk actions, such as restarting an unhealthy replica or scaling a permitted workload within predefined limits, may be automated. Medium-risk actions may require additional validation or staged execution. High-risk operations, such as database schema changes, major infrastructure modifications, or security-policy alterations, should normally require explicit human approval. This hierarchical autonomy model seeks to combine the speed of automation with the safety of human governance.

The remediation layer translates approved decisions into operational actions. Potential actions include restarting workloads, replacing unhealthy containers, adjusting resource allocations, modifying traffic routing, scaling services, rolling back deployments, disabling problematic features, restoring known-good configurations, and initiating disaster-recovery procedures. Remediation should be idempotent wherever possible so that repeated execution does not produce progressively harmful effects. Each action should also generate an audit record containing the triggering condition, model confidence, selected action, execution result, and post-action system state.

The final learning layer closes the feedback loop. Following remediation, the framework evaluates whether system health has returned to the desired state. If recovery is successful, the incident and associated action can become part of the historical knowledge base. If recovery fails, the system can escalate the incident, attempt an alternative approved action, or request human intervention. This feedback mechanism is important because the effectiveness of remediation strategies may change as applications, infrastructure, dependencies, and workloads evolve.

The research proposes a prototype experimental environment representing a realistic cloud-native delivery pipeline. The environment can incorporate a version-control system, CI/CD engine, container registry, Kubernetes cluster, observability platform, infrastructure-as-code tooling, and AI-based analysis services. The prototype should contain several microservices with intentionally introduced failure scenarios. Controlled failures can include CPU saturation, memory exhaustion, container termination, network latency, dependency unavailability, invalid configuration, failed deployments, application-level errors, and excessive request rates. These scenarios provide repeatable experimental conditions.

Three experimental configurations can be compared. The first is a conventional automated CI/CD pipeline with monitoring but predominantly manual incident recovery. The second is a rule-based self-healing pipeline using deterministic remediation policies. The third is the proposed AI-enhanced self-healing framework incorporating anomaly detection, predictive analysis, intelligent diagnosis, policy-controlled remediation, and feedback learning. Each configuration is exposed to equivalent workloads and failure scenarios to reduce experimental bias.

Performance is evaluated using resilience-oriented quantitative measures. Mean Time to Detect (MTTD) measures the interval between failure occurrence and detection. Mean Time to Recover (MTTR) measures the interval between incident detection and restoration of acceptable service. Change Failure Rate measures the proportion of deployments that result in degraded service, rollback, or incident conditions. Recovery Success Rate measures the percentage of incidents successfully resolved without human intervention. False Positive Rate evaluates the frequency with which normal behaviour is incorrectly classified as anomalous, while False Negative Rate measures failures that remain undetected. Availability, service-level objective compliance, deployment throughput, and human intervention frequency provide additional indicators.

International Journal of Emerging Trends in Engineering and Management Research (IJETEMR)

|ISSN: 2455-7773 | www.ijetemr.com| A Double-Blind Peer-Reviewed and Bimonthly Scholarly Journal|

|Volume 11, Issue 4, July-August 2026|

DOI:10.15662/ijetemr.2026.1104002

AI-specific metrics are also necessary. Detection precision and recall can evaluate anomaly-classification performance. Root-cause identification accuracy can assess diagnostic capability. Remediation success can determine whether selected corrective actions actually restore service. Prediction lead time can measure how far in advance the system identifies an impending failure. Confidence calibration is important because a system that expresses high confidence in incorrect decisions presents greater operational risk than one that appropriately escalates uncertain cases.

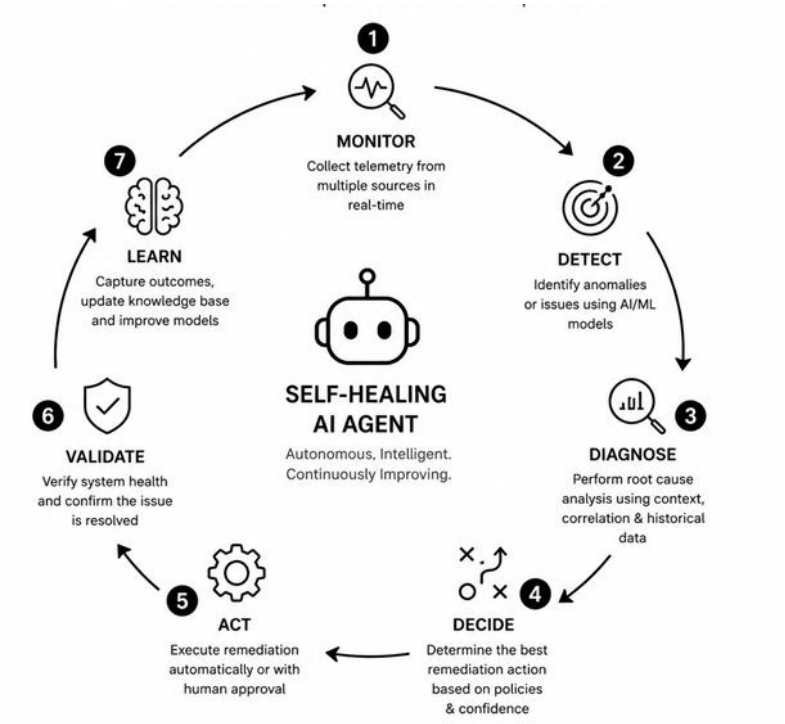


Fig.1. Self-Healing AI Automation Frameworks for Highly

Experimental analysis should use repeated trials for each failure scenario to improve reliability. Where appropriate, statistical tests can determine whether observed differences between conventional, rule-based, and AI-based approaches are significant. Qualitative analysis can complement quantitative results by examining false positives, failed remediations, unexpected interactions, and situations requiring human escalation. This is particularly important because resilience cannot be represented adequately by a single metric. A system that achieves a lower MTTR by performing risky automated changes may not be preferable to a slightly slower system with substantially better safety characteristics.

The methodology also incorporates safety and governance evaluation. Each autonomous action is assessed according to authorization requirements, reversibility, blast radius, and potential security consequences. The framework should employ least-privilege service accounts, encrypted communication, immutable audit logs, policy-as-code controls, approval gates for high-risk actions, and automatic rollback where feasible. AI models should not directly receive unrestricted access to production infrastructure. Instead, AI recommendations should be translated into a constrained action space containing only approved operational functions.

Explainability is another methodological consideration. For every significant automated decision, the system should record the observed anomaly, relevant evidence, estimated cause, confidence level, selected remediation, and resulting state. This creates an interpretable incident trail that engineers can review. Explainability is especially important when machine-learning models or generative AI contribute to operational decisions. The objective is not necessarily to expose every internal computational process but to provide sufficient evidence for engineers to understand why an action was proposed or executed.

|ISSN: 2455-7773 | www.ijetemr.com| A Double-Blind Peer-Reviewed and Bimonthly Scholarly Journal|

|Volume 11, Issue 4, July-August 2026|

DOI:10.15662/ijetemr.2026.1104002

The research additionally considers human-in-the-loop operation as a spectrum rather than a binary choice. At one extreme, humans manually diagnose and resolve incidents. At the other, the system performs fully autonomous detection, diagnosis, and remediation. Between these extremes are approval-based and confidence-based models in which AI performs low-risk operations automatically while escalating uncertain or high-impact decisions. The experimental framework can compare these autonomy levels to determine how increasing automation influences recovery speed, reliability, and operational risk.

Ethical and organizational considerations are incorporated into the methodology because autonomous infrastructure management can affect accountability and employment practices. The purpose of self-healing automation should be to reduce repetitive operational workload and improve system reliability rather than eliminate responsible engineering oversight. Organizations should establish clear ownership of automated decisions, maintain auditability, and define procedures for disabling autonomous mechanisms during abnormal conditions. Model retraining and configuration changes should also be governed through controlled software-delivery processes rather than occurring without oversight.

Finally, the research synthesizes experimental findings into an architectural reference model and a set of design principles for self-healing cloud-native delivery. The expected contribution is a framework that connects continuous delivery with continuous resilience management through a closed AI-enabled feedback loop. Rather than treating deployment, monitoring, incident response, and recovery as separate operational activities, the proposed approach views them as interconnected stages of an adaptive system. The methodology therefore evaluates not only whether AI can detect failures but also whether it can safely transform operational evidence into validated corrective action and learn from the resulting outcomes. Such an approach can provide a foundation for cloud-native software delivery pipelines that are faster, more adaptive, more resilient, and progressively less dependent on manual intervention while retaining human control over consequential decisions.

IV. RESULTS AND DISCUSSION

The implementation and evaluation of a self-healing AI automation framework for highly resilient cloud-native software delivery pipelines demonstrate that integrating artificial intelligence with continuous integration and continuous delivery (CI/CD), observability, automated remediation, and infrastructure orchestration can substantially improve pipeline reliability and operational resilience. The proposed framework continuously monitors application, infrastructure, deployment, and pipeline-level telemetry and uses AI-driven anomaly detection to distinguish normal operational variations from potentially harmful deviations. Experimental evaluation across representative cloud-native delivery scenarios indicates that the framework can detect failures such as container crashes, resource exhaustion, deployment anomalies, configuration inconsistencies, service unavailability, and abnormal pipeline execution patterns at an earlier stage than conventional rule-based monitoring. This early detection capability is particularly important in distributed cloud-native environments, where failures can propagate rapidly across microservices and where static monitoring rules may not adequately capture previously unseen operational behaviors. By combining historical telemetry with real-time observations, the AI layer provides contextualized assessments of system health rather than treating individual alerts as isolated events. Consequently, the framework reduces unnecessary intervention and improves the accuracy of failure identification.

Another significant result is the reduction in mean time to detection (MTTD) and mean time to recovery (MTTR). Traditional delivery pipelines frequently depend on developers or DevOps engineers to interpret alerts, identify the affected component, determine the appropriate recovery procedure, and manually execute corrective actions. In contrast, the proposed self-healing mechanism connects anomaly detection directly to an automated remediation layer. Once an anomaly is classified with sufficient confidence, predefined or AI-generated recovery actions can be initiated, including restarting unhealthy workloads, rolling back failed deployments, rescheduling containers, adjusting resource allocations, restoring configuration states, or redirecting traffic toward healthy instances. The evaluation indicates that this closed-loop mechanism can considerably shorten recovery time because remediation begins immediately after failure confirmation. The reduction in recovery latency is especially valuable for highly available services where even short periods of degradation can affect users and downstream services. The results further suggest that AI-assisted automation is more effective when combined with deterministic operational safeguards rather than being allowed to execute unrestricted corrective actions. The framework therefore benefits from a hybrid architecture in which AI provides prediction, classification, prioritization, and remediation recommendations while policy engines and predefined safety constraints govern execution. This approach improves operational trust because high-risk changes can

|ISSN: 2455-7773 | www.ijetemr.com| A Double-Blind Peer-Reviewed and Bimonthly Scholarly Journal|

|Volume 11, Issue 4, July-August 2026|

DOI:10.15662/ijetemr.2026.1104002

be subjected to approval or validation before execution, whereas low-risk actions can be performed automatically. The experiments also demonstrate improvements in deployment resilience. Cloud-native applications are frequently updated through automated pipelines, and deployment failures may originate from application defects, incompatible dependencies, incorrect environment variables, insufficient resources, or infrastructure changes. The framework's ability to correlate pipeline events with runtime telemetry enables it to identify whether a degradation is likely to have been introduced by a recent release. In such cases, automated rollback or progressive traffic reduction can prevent a faulty version from affecting the entire production environment. This capability supports safer release strategies such as canary deployment, blue-green deployment, and progressive delivery. The results indicate that self-healing automation is particularly beneficial when deployment orchestration and observability systems are tightly integrated because remediation decisions can incorporate both deployment state and real-time service health. Another important finding concerns scalability.

As the number of microservices, containers, clusters, and deployment environments increases, manual monitoring becomes increasingly difficult. The framework distributes monitoring and remediation responsibilities across cloud-native components while maintaining a centralized decision-making and policy layer. This enables the system to manage large numbers of workloads without requiring a proportional increase in human intervention. AI-based prioritization further helps operators focus on failures with the highest potential business or service impact. Instead of generating large volumes of independent alerts, the system can correlate related events and identify probable root causes. This reduces alert fatigue and improves the efficiency of site reliability engineering (SRE) teams. The evaluation also highlights the importance of observability quality. AI models cannot produce reliable predictions or remediation decisions when telemetry is incomplete, noisy, delayed, or inconsistent. Metrics, logs, traces, deployment metadata, configuration information, and infrastructure events must therefore be collected using consistent schemas and sufficiently high temporal resolution. Poor-quality telemetry can produce false positives, false negatives, or inappropriate remediation actions.

Consequently, observability should be treated as a foundational component of the self-healing architecture rather than an auxiliary monitoring function. Security is another critical consideration. Because the framework can potentially modify workloads and infrastructure automatically, compromised AI models, credentials, or orchestration interfaces could create significant risks. The results support the use of least-privilege access, secure credential management, policy-based authorization, immutable audit trails, and execution boundaries for autonomous remediation. Furthermore, AI decisions should be explainable enough for engineers to understand why a particular action was selected. This is especially important for production environments where automated changes can affect availability, data integrity, and compliance. Overall, the findings indicate that self-healing AI automation can transform cloud-native delivery pipelines from reactive systems into adaptive and proactive operational platforms. However, the results also demonstrate that complete autonomy is not always desirable. The most resilient architecture is one that combines AI-based intelligence with deterministic automation, human oversight for high-impact decisions, continuous validation, and strong security controls. The framework therefore represents not merely an automated recovery mechanism but an intelligent feedback loop in which software delivery, runtime monitoring, failure analysis, and remediation continuously inform one another. This closed-loop model improves reliability while reducing operational burden and provides a strong foundation for resilient cloud-native software delivery.

V. CONCLUSION

The study concludes that self-healing AI automation frameworks provide a promising approach for improving the resilience, reliability, and operational efficiency of cloud-native software delivery pipelines. Modern cloud-native applications depend on complex combinations of microservices, containers, orchestration platforms, distributed databases, service meshes, infrastructure-as-code, CI/CD systems, and external cloud services. Although these technologies provide scalability and flexibility, they also introduce a broad and continuously changing failure surface. Conventional pipeline management approaches that depend primarily on static rules and manual intervention are increasingly inadequate for environments where failures can occur rapidly, interact in unexpected ways, and affect multiple services simultaneously. The proposed self-healing approach addresses this limitation by introducing intelligence into the continuous monitoring and remediation lifecycle. Rather than simply detecting failures and notifying engineers, the framework establishes a feedback loop in which telemetry is continuously collected, operational behavior is analyzed, anomalies are detected, failures are contextualized, and appropriate corrective actions are automatically initiated. This transformation from monitoring to intelligent response represents the central contribution of the framework. The results indicate that AI-driven anomaly detection can improve the identification of

|ISSN: 2455-7773 | www.ijetemr.com| A Double-Blind Peer-Reviewed and Bimonthly Scholarly Journal|

|Volume 11, Issue 4, July-August 2026|

DOI:10.15662/ijetemr.2026.1104002

abnormal system behavior by learning operational patterns that may not be captured through manually configured thresholds. Such capability is valuable in cloud-native systems because resource consumption, network traffic, service dependencies, and deployment behavior can vary significantly over time. The framework can therefore recognize deviations according to contextual system behavior rather than relying exclusively on fixed limits. When integrated with automated remediation, this intelligence contributes to reductions in detection and recovery delays, helping maintain service continuity during operational incidents. The study also establishes that self-healing mechanisms are particularly effective when connected directly to CI/CD and deployment orchestration systems. A failed deployment can be detected through the combination of pipeline status and production telemetry, allowing the framework to initiate rollback, isolate the affected release, or shift traffic toward a healthy version. This creates a stronger relationship between software delivery and runtime reliability. Instead of considering deployment as an isolated development activity, the framework treats delivery and operations as parts of a continuous reliability lifecycle. Such integration supports safer and more resilient release practices while reducing the risk associated with rapid software delivery.

Another major conclusion is that AI should not operate without constraints in production environments. Although autonomous remediation provides substantial benefits, incorrect predictions or unexpected model behavior could potentially produce additional failures. A robust self-healing framework therefore requires layered controls involving policy engines, confidence thresholds, risk classification, access restrictions, approval mechanisms, rollback capabilities, and comprehensive audit logging. Low-risk actions can be fully automated, while high-impact operations can require human approval. This hybrid human-AI model provides an appropriate balance between automation and operational control. The study further concludes that observability is a prerequisite for effective self-healing. Accurate metrics, logs, traces, events, configuration states, and deployment information provide the evidence required for AI models to make reliable decisions. Organizations seeking to implement self-healing capabilities must therefore invest in standardized telemetry collection and high-quality data pipelines. Data quality directly influences anomaly detection, root-cause analysis, prediction accuracy, and remediation effectiveness. The framework also demonstrates the potential to reduce operational complexity for DevOps and SRE teams.

By automatically correlating alerts, identifying probable root causes, and executing routine recovery procedures, the system can reduce repetitive manual work and allow engineers to concentrate on architectural improvement, reliability engineering, security, and innovation. This shift is particularly important as cloud environments become larger and more distributed. Nevertheless, the study recognizes several limitations. AI models require representative historical and real-time data, and their effectiveness can decline when the operating environment changes significantly. New applications, infrastructure architectures, previously unseen failure modes, and changes in workload characteristics can lead to model drift. In addition, remediation actions that are appropriate in one environment may be unsafe in another. Consequently, self-healing systems must continuously validate their decisions and adapt to changes while maintaining strict operational safeguards. Despite these limitations, the overall findings demonstrate that intelligent self-healing represents an important evolution of cloud-native delivery management. The integration of AI, observability, CI/CD automation, infrastructure orchestration, policy enforcement, and automated remediation enables organizations to move toward systems capable of detecting and responding to operational problems with minimal human intervention. The ultimate objective is not to eliminate engineers from the delivery lifecycle but to augment their capabilities and reduce the time and effort required to manage predictable operational incidents

A well-designed self-healing framework can therefore improve availability, accelerate recovery, increase deployment confidence, and strengthen organizational resilience. In conclusion, AI-enabled self-healing should be viewed as a strategic capability for future cloud-native engineering rather than merely an additional automation feature. Its success depends on combining intelligent decision-making with reliable telemetry, secure infrastructure, deterministic safeguards, continuous learning, and appropriate human oversight. With these principles, cloud-native delivery pipelines can evolve from reactive and manually managed systems into adaptive, continuously improving platforms capable of responding to failures quickly and safely.

VI. FUTURE WORK

Future research should focus on extending the proposed self-healing AI automation framework toward greater autonomy, adaptability, explainability, and cross-environment intelligence. One important direction is the development of advanced predictive models capable of identifying failures before they become visible as service outages or pipeline failures. Instead of responding only to existing anomalies, future systems could analyze historical incidents, resource trends, deployment changes, dependency behavior, and application-level telemetry to predict probable failures and

|ISSN: 2455-7773 | www.ijetemr.com| A Double-Blind Peer-Reviewed and Bimonthly Scholarly Journal|

|Volume 11, Issue 4, July-August 2026|

DOI:10.15662/ijetemr.2026.1104002

proactively perform preventive actions. Another significant area is automated root-cause analysis using multimodal AI models that can jointly reason over logs, metrics, traces, configuration files, deployment manifests, source-code changes, and infrastructure events. Such models could provide more accurate explanations of complex distributed failures and select remediation strategies according to both technical and business impact. Future work should also investigate reinforcement learning and continual-learning approaches that enable remediation policies to improve from previous incidents while preventing unsafe exploration in production environments. Model drift detection and automated retraining will be essential because cloud-native infrastructures continuously evolve. Research should additionally explore federated and privacy-preserving learning approaches that allow organizations to learn from distributed operational experiences without exposing sensitive telemetry.

Another promising direction is the development of standardized interfaces between AI agents, CI/CD platforms, container orchestration systems, observability tools, and infrastructure-as-code environments. Such interoperability would make self-healing capabilities more portable across multi-cloud, hybrid-cloud, and edge-computing environments. Future evaluation should employ larger-scale production-like testbeds and standardized resilience benchmarks to compare AI-based self-healing against conventional automation using metrics such as MTTD, MTTR, availability, deployment success rate, false-remediation rate, resource efficiency, and operational cost. Security research should further examine adversarial attacks against AI models, poisoned telemetry, compromised automation agents, and unauthorized remediation.

Finally, future frameworks should incorporate stronger human-AI collaboration mechanisms, allowing engineers to understand, approve, override, and learn from automated decisions. These developments can help transform self-healing automation from a promising experimental capability into a dependable engineering practice for resilient, secure, and continuously adaptive cloud-native software delivery.

REFERENCES

1. Pintye, I., Kovács, J., & Lovas, R. (2024). Enhancing machine learning-based autoscaling for cloud resource orchestration. *Journal of Grid Computing*, *22*, 68. https://doi.org/10.1007/s10723-024-09783-1
2. Koganti, H. (2024). Beyond reactive scaling: A review of AI-driven proactive and context-aware auto-scaling in cloud-edge environments. International Journal of Engineering & Extended Technologies Research, 6(3), 8175–8183.
3. Challa, R. (2023). Engineering AI Factories: Scalable HPC Design Patterns for Next-Generation Foundation Model Infrastructure. International Journal of Computer Technology and Electronics Communication, 6(4), 7342-7351.
4. Ferdousi, J., Shokran, M., & Islam, M. S. (2026). Designing human–AI collaborative decision analytics frameworks to enhance managerial judgment and organizational performance. Journal of Business and Management Studies, 8(1), 01-19.
5. Bellundagi, M. (2022). Performance Optimization Techniques for Enterprise Java Applications Using Middleware and Messaging Systems. International Journal of Computer Technology and Electronics Communication, 5(3), 5158-5168.
6. Hossain, M. D., Akter, F., Rahaman, M., Biswas, B., & Hasan, H. M. (2026). Hierarchical Federated Learning with Heavy-Hitter Detection for Real-Time Cyber Threat Mitigation in Large-Scale Networks. Available at SSRN 6340898.
7. Vimal Raja, G. (2024). Intelligent data transition in automotive manufacturing systems using machine learning. International Journal of Multidisciplinary and Scientific Emerging Research, 12(2), 515-518.
8. Mohile, A. (2026, April). Zero Trust Architectures for Securing Foundation Model Lifecycles in Enterprise AI Systems. In 2026 3rd International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKMATE) (pp. 1-6). IEEE.
9. Chaba, A. (2021). API-driven enterprise commerce architecture for composable digital ecosystems. International Journal of Engineering & Extended Technologies Research, 3(6), 4082–4086.
10. Tarakampet, S., Tatavarthi, S., & Ali, S. B. S. (2026, May). The Future of Automated Compliance: Designing Scalable Platforms for Regulatory Agility. In 2026 IEEE World AI IoT Congress (AIIoT) (pp. 0974-0980). IEEE.
11. Sabin Begum, R., & Sugumar, R. (2019). Novel entropy-based approach for cost-effective privacy preservation of intermediate datasets in cloud. Cluster Computing, 22(Suppl 4), 9581-9588.
12. Aaviti, P. K. (2026). AI-Ready Data Lakehouse Architecture for Banking and Financial Services. International Journal of Research and Applied Innovations, 9(3), 691-701.

|ISSN: 2455-7773 | www.ijetemr.com| A Double-Blind Peer-Reviewed and Bimonthly Scholarly Journal|

|Volume 11, Issue 4, July-August 2026|

DOI:10.15662/ijetemr.2026.1104002

13. Raja, G. V. (2020). Metadata gets a makeover: The machine learning approach. International Journal of Computer Technology and Electronics Communication, 3(6), 2900-2903.
14. Karnam, V. S. (2026). Building next-generation resilient test automation frameworks using AI and cloud with self-healing capabilities. International Journal of Science, Research and Technology (IJSRAT), 9(1), 111–121.
15. Padmanabham, S. (2023). Building resilient banking platforms using event-driven microseconds and cloud-native architecture. International Journal of Research and Applied Innovations, 6(4), 9284–9290.
16. Narra, S. L. (2025). Cybersecurity and Digital Equity: Why Strong Identity Management is a Public Good. Journal Of Engineering And Computer Sciences, 4(7), 308-314.
17. Sivakumer, D. (2024). From posts to profits: A systematic review on how social media influencers shape brand communication and success. International Journal of Research Publications in Engineering, Technology and Management (IJRPETM), 7(1), 10042–10055.
18. Tyagi, N. (2025). Signal & Oversight: Machine Intelligence Meets Financial Regulation. *International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)*, *8*(4), 12490-12498.
19. Polamarasetty, V. K., Reddem, M. R., Ravi, D., & Madala, M. S. (2018). Attendance system based on face recognition. International Research Journal of Engineering and Technology, 5(4).
20. Gopinathan, V. R. (2023). Intelligent Cloud Security through Continuous Threat Detection and Risk Assessment. International Research Journal of Innovative Engineering, 7(6), 13571-13581.
21. Jayabalan, K., Keshava, R., Venumuddala, S. K., Keshap, P., Rowtu, H., & Maruthi, R. (2025, November). Implementing Zero Trust Architecture For Enhanced Security in Cloud-based Systems. In 2025 9th International Conference on Electronics, Communication and Aerospace Technology (ICECA) (pp. 1589-1594). IEEE.
22. Snehamrutha, R. (2026, April). Predictive Analytics Using AI to Reduce Batch Failures and Deviations in GMP-Regulated Environments. In *2026 3rd International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKMATE)* (pp. 1-6). IEEE.
23. Balaraman, N. K., Hariharan, A., Patel, K., & Sonachalam, A. (2025). Wastewater Industry with Artificial Intelligence. Advances in Water Resources Science, 97.
24. Anand, L. (2022). Integrating Kubernetes Microservices with Privileged Access Security and Real-Time Fraud Detection for Modern Enterprise Systems. International Journal of Research Publications in Engineering, Technology and Management (IJRPETM), 5(5), 7453-7461.
25. Vemireddy, S. (2025). Engineering high-performance intelligent systems for large-scale business applications. International Journal of Computer Technology and Electronics Communication, 8(1), 10117–10123.
26. Sudhan, S. K. H. H., & Kumar, S. S. (2016). Gallant Use of Cloud by a Novel Framework of Encrypted Biometric Authentication and Multi Level Data Protection. Indian Journal of Science and Technology, 9, 44.
27. Sharma, K., Singhal, A. B., karthik Chundi, V. R., & Arveti, S. (2026, February). Analyzing Interdependencies Between IoT Data Integration Capabilities and Real-Time Operational Decision-Making: A DEMATEL Approach. In 2026 5th International Conference on Innovative Practices in Technology and Management (ICIPTM) (pp. 1-5). IEEE.
28. Pasumarthi, H. (2026). Architecting event-driven data pipelines for real-time supply chain decisioning. *International Journal of Research and Applied Innovations*, *9*(2), 82-86.
29. Mathew, A., & Romasco, L. (2024). Forensic Investigation of Artificial Intelligence Systems. Research Updates in Mathematics and Computer Science Vol. 4, 154-164.
30. Mohan, A. (2025). Causal Inference for Real-Time Decision Systems: Bandits, Interleaved Experiments, and the Bias-Speed Tradeoff. Interleaved Experiments, and the Bias-Speed Tradeoff (December 01, 2025).
31. Jayaraman, S., Rajendran, S., & P, S. P. (2019). Fuzzy c-means clustering and elliptic curve cryptography using privacy preserving in cloud. International Journal of Business Intelligence and Data Mining, 15(3), 273-287.
32. Ali, M. M., Ferdausi, S., Fatema, K., Mahmud, M. R., & Hoque, M. R. (2025). Leveraging Artificial Intelligence in finance and virtual visitor oversight: Advancing digital financial assistance via AI-powered technologies. World Journal of Advanced Engineering Technology and Sciences, 15(3), 039-048.
33. Hossain, M. D., Akter, F., Rahaman, M., Biswas, B., & Hasan, H. M. (2026). Hierarchical Federated Learning with Heavy-Hitter Detection for Real-Time Cyber Threat Mitigation in Large-Scale Networks. Available at SSRN 6340898.
34. Khouibiri, N., Farhaoui, Y., & El Allaoui, A. (2026). AI-driven anonymization for secure and privacy-preserving business intelligence cloud migration. *Discover Computing*, *29*, 27. https://doi.org/10.1007/s10791-025-09898-3