Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Self-Healing Infrastructure: Leveraging Reinforcement Learning for Autonomous Cloud Recovery and Enhanced Resilience

Rohit Laheri[1], Harish Kumar.Krishnamurthy.Sukumar[2], Chandrashekar Kola[3], Yashasvi Makin[4]
[1]*Software Engineer, Tech Mahindra, Texas, USA*
[2]*Associate Principal - Cloud Engineering, LTIMindtree, Texas, USA*
[3]*Senior Systems Engineer, Autozone Inc, Tennessee, USA*
[4]*Software Engineer, Seattle, WA*

ARTICLE INFO

Received: 21 Mar 2025

Revised: 30 Apr 2025

Accepted: 17 May 2025

ABSTRACT

Maintaining high availability and reliability in dynamic cloud environments demands proactive, automated solutions capable of handling failures at scale. This study introduces a novel multi-layer self-healing infrastructure framework that unites predictive analytics, reinforcement learning (RL), and rule-based automation into a cohesive, horizontally scalable system. Predictive analytics continuously ingests telemetry—CPU, memory, network metrics, and application logs—using time-series forecasting (ARIMA, LSTM) and unsupervised anomaly detection (Isolation Forest, k-Means) to flag potential faults with >96% accuracy. An RL agent employing Proximal Policy Optimization (PPO) then dynamically selects recovery actions (e.g., container restart, horizontal scaling, resource reallocation) guided by a reward function that balances rapid Mean Time To Repair (MTTR) reduction with minimal resource overhead and service impact. Simultaneously, rule-based playbooks address frequent failure patterns, ensuring immediate remediation within 30 seconds for predictable incidents. Deployed as Infrastructure as Code (IaC) via Terraform and Helm on Kubernetes clusters across AWS and Azure, our framework was validated over 220 fault scenarios. Key performance indicators demonstrate an 85% MTTR reduction (from 90 to 13.5 minutes), recovery reliability exceeding 95%, fault tolerance above 91%, and system uptime surpassing 98%. Resource overhead during recovery remains under 10%. Compared to prior isolated methods—rule-based MTTR reduction of 60%, RL-only MTTR reduction of 50%, and anomaly detection without remediation—our integrated model delivers superior resilience and operational efficiency. This paper is organized as follows: Section 1 introduces the problem and contributions; Section 2 reviews related work; Section 3 details the methodology; Section 4 describes experimental setup; Section 5 presents results; Section 6 discusses implications and limitations; Section 7 outlines future research directions; and Section 8 concludes.

Keywords: self-healing, cloud resilience, reinforcement learning, predictive analytics, rule-based automation, infrastructure as code, Kubernetes, cloud recovery.

1. Introduction

Cloud computing has revolutionized enterprise IT by offering on-demand scalability, elasticity, and cost savings. Modern architectures built on microservices and container orchestration enable rapid feature deployment but also introduce complexity in maintaining service reliability. Hardware failures, software bugs, misconfigurations, and network anomalies can propagate through interdependent components, causing cascading outages. Industry reports indicate an average of 14 hours of annual downtime per organization, costing roughly $5,600 per minute in lost productivity and revenue (Gartner, 2023). Translating to a mid-size enterprise, this equates to over $4 million annually in potential losses.

Traditional recovery approaches rely on threshold-based alerts and manual interventions or static remediation scripts. While effective for common failure modes, these practices suffer from delayed detection-to-response times,

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

fragmented toolchains, and significant operational overhead. The gap between alert generation and action often results in prolonged MTTR, eroding service-level agreements and user trust.
Self-healing systems aim to automate detection, diagnosis, and remediation tasks, minimizing human involvement. Early autonomic computing research (Kephart & Chess, 2003) inspired the development of event-driven automation engines, yet these systems lack adaptability to novel fault patterns. Predictive analytics and forecasting provide early warnings, but without integrated remediation, they leave operators overwhelmed. Reinforcement learning has demonstrated promise in dynamic decision-making for recovery, but training complexity and cross-environment policy transfer remain challenges. Furthermore, most frameworks focus on a single paradigm, limiting their end-to-end efficacy.

To address these challenges, we propose a unified, multi-layer self-healing framework that combines:

1. **Proactive Anomaly Detection:** High-accuracy forecasting and unsupervised detection to anticipate failures.
2. **Adaptive RL Decision-Making:** PPO-based agent optimizing recovery actions under evolving conditions with exploration-exploitation balance via entropy bonuses.
3. **Deterministic Rule Execution:** Immediate playbook-driven responses for routine incidents.
4. **IaC-Enabled Deployment:** Terraform and Helm for consistent, secure multi-cloud rollout with Vault-managed secrets.

Our objectives are to (i) minimize MTTR, (ii) maximize recovery reliability and fault tolerance, and (iii) validate cross-platform scalability and operational feasibility.

2. Literature Review

Research in self-healing cloud infrastructure encompasses three primary paradigms. Event-driven automation extends the autonomic computing "if-this-then-that" model by correlating metrics, logs, and topology to select deterministic remediation scripts. Though low-latency for known faults, such systems require continuous rule maintenance and fail with novel or compound failures. Predictive analytics leverages ARIMA and LSTM models to forecast metric thresholds and employs Isolation Forest and clustering to uncover anomalies without labels, achieving detection accuracy above 94%. However, these methods traditionally stop at alerting, leading to alert fatigue and no automated recovery. Reinforcement learning frames recovery as a Markov Decision Process, where agents learn optimal remediation policies through trial-and-error simulations. PPO algorithms exhibit stable policy convergence within ~250 episodes, reducing MTTR by up to 50% compared to static approaches. Yet RL demands extensive synthetic fault injection, and policy generalization across heterogeneous environments often underperforms without fine-tuning. Platform-centric diagnostic pipelines integrate anomaly detection with root cause analysis via graph-based dependency mapping, streamlining post-incident investigations but lacking automated remediation. A comparative summary (Table 1) highlights that no single approach simultaneously addresses proactive detection, adaptive remediation, and rapid automation at scale. Our work synthesizes these strengths into a cohesive architecture that fills this gap.

Table 1. Comparative Overview of Self-Healing Paradigms

Paradigm	Detection	Decision-Making	Remediation	Limitations
Event-Driven Automation	Threshold-based	Rule-based	Script execution	Non-adaptive to novel faults
Predictive Analytics	Time-series	N/A	N/A	No integrated remediation
Reinforcement Learning	Reactive	Adaptive (PPO)	Learned policies	High training overhead
Diagnostic Pipelines	Unsupervised	N/A	N/A	Alerts only, no automated recovery
Integrated Framework	Proactive & Anomaly	Adaptive & Rule-based	Layered remediation	Balanced complexity and scalability

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

3. Methodology

3.1 Anomaly Detection

We collect real-time telemetry data via Prometheus and Fluentd, scraping over 150 distinct metrics at 15-second intervals. Metrics include node-level CPU utilization, memory usage, disk I/O, network throughput, garbage collection durations, and application-specific health probes. Logs are aggregated in Elasticsearch, linked to metrics via unique request identifiers. Raw data is retained for 90 days to support both near-term diagnostics and long-term trend analysis.

Feature Engineering:

- **Statistical Descriptors**: For each metric over a rolling one-minute window, we calculate mean, median, variance, skewness, and kurtosis to capture distributional characteristics.
- **Spectral Analysis**: Fast Fourier Transform (FFT) is applied to sliding windows to extract dominant frequency components, revealing periodic workload cycles and oscillations.

Anomaly Modeling:

- **Isolation Forest**: Detects extreme outliers by building an ensemble of decision trees, flagging data points with an anomaly score above 0.75. The model is retrained weekly with a contamination parameter of 1%.
- **k-Means Clustering**: Groups operational states into clusters; points falling outside a Mahalanobis distance threshold of 3 from any centroid are labeled as anomalies.

A voting ensemble combines model outputs into a composite anomaly score, which is smoothed using an Exponentially Weighted Moving Average (EWMA) to filter transient spikes. Sustained scores above the dynamic threshold trigger the next remediation layer.

3.2 Time-Series Forecasting

To anticipate failures, we implement two forecasting models:

- **ARIMA**: Configured with parameters (p=5, d=1, q=2) to account for autoregression, differencing, and moving average components. Regular retraining on 30-day sliding windows maintains model relevance.
- **LSTM Neural Network**: Consists of two LSTM layers (64 and 32 units) followed by a dense output, trained on multivariate sequences of five-minute summaries. Dropout regularization (0.2) reduces overfitting.

Forecasts are generated for 5–10 minute horizons. Model performance is evaluated via Mean Absolute Percentage Error (MAPE), with ARIMA achieving 6% and LSTM 4%. Forecasted values that exceed SLA thresholds by more than 5% raise preemptive alerts.

3.3 Reinforcement Learning for Adaptive Recovery

MDP Design

- **State Representation**: A 50-dimensional vector combining: current anomaly score, latest forecast deviations, system load averages, and service latency percentiles.
- **Action Set**: Discrete actions include: restart a failing container, horizontally scale a deployment by ±1 replica, cordon and drain a faulty node, and adjust resource requests (CPU/memory).
- **Reward Signal**: Defined as: $r = \alpha \times \Delta(\text{time to stability}) - \beta \times (\text{additional resource usage}) - \gamma \times (\text{service impact penalty})$, where $\alpha=1.0$, $\beta=0.5$, $\gamma=0.3$. Stability is measured by the return of latency and error rates to baseline ranges.

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Training and Policy Deployment

We leverage Proximal Policy Optimization (PPO) with an entropy coefficient of 0.01 to encourage exploration. Training uses synthetic fault injection in sandbox clusters for 300 episodes, each lasting up to five minutes or until stability is achieved. Policies converge at an average of 250 episodes, with training durations of 3–4 hours on a 16 vCPU, 64 GB RAM node. Trained models are containerized and deployed as Kubernetes sidecar pods, communicating actions via the Kubernetes API.

3.4 Rule-Based Automation

We maintain a library of 60 remediation playbooks in YAML format, each containing:

- **Trigger Conditions**: Kubernetes events (e.g., CrashLoopBackOff, OOMKilled) or Prometheus alerts.
- **Precondition Validations**: Checks for node readiness, pod health endpoints, and dependency availability.
- **Remediation Sequence**: Commands executed sequentially, such as scaling commands, config reloads, or service restarts.

A lightweight orchestration engine, implemented in Go, listens to the Kubernetes API server and triggers playbooks with an average execution latency under 30 seconds, ensuring rapid resolution of common faults.

3.5 IaC Deployment and Security

All resources are declared via Terraform modules, including VPC configurations, node pools, IAM roles, and network policies. Helm charts manage application components: anomaly detectors, RL agents, and automation controllers. We integrate HashiCorp Vault for dynamic secrets management, enabling secure retrieval of database credentials, API tokens, and TLS certificates at runtime. Continuous deployment is orchestrated by GitHub Actions, which run policy-as-code checks (using tfsec and kube-linter) before applying IaC changes. Argo CD monitors cluster state, enforcing GitOps workflows and automatically rolling back changes upon drift detection.

4. Experimental Setup

4.1 Cluster Configuration

We provision two identical clusters (AWS EKS, Azure AKS) with 5 nodes (4 vCPU, 16 GB RAM) each, hosting a 10-service e-commerce microservices application.

4.2 Fault Injection

Using Chaos Mesh, we inject 220 randomized faults over 30 days: hardware reboots, disk latency, memory leaks, CPU spikes, network latency (5–20%), and DNS failures.

4.3 Baseline Comparison

Baseline recovery leverages static remediation scripts and manual intervention. Metrics are captured via Prometheus logs and aligned with incident timestamps for MTTR measurement.

5. Results

5.1 KPI Outcomes

Table 2. Performance Comparison

Metric	Baseline	Proposed	Delta
MTTR (min)	90	13.5	-85%
Recovery Reliability (%)	82.3	95.7	+13.4 pp
Fault Tolerance (%)	68.1	91.3	+23.2 pp
System Uptime (%)	92.4	98.7	+6.3 pp
Overhead (CPU/Mem %)	3.2/2.1	8.7/3.8	+5.5/1.7 pp

5.2 Fault-Type Breakdown

Journal of Information Systems Engineering and Management
2025, 10(49s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Table 3. Fault Category Insights

Category	Count	Avg. MTTR	Reliability (%)	Tolerance (%)
Hardware	70	12.2	96.4	94.3
Software	80	15.8	93.8	89.9
Network	50	13.0	95.0	90.0

5.3 Sensitivity Analysis

Adjusting anomaly thresholds by ±10% impacts MTTR by ±5%, demonstrating robustness in early detection calibration.

6. Discussion

Our layered framework outperforms isolated paradigms by harmonizing proactive detection, adaptive RL, and deterministic automation. Compared to rule-based MTTR reduction of 60% and RL-only reduction of 50% reported in prior work, our integrated model achieves 85% reduction. IaC deployment ensures consistent cross-cloud performance. However, RL training overhead and security-sensitive fault handling remain areas for improvement.

7. Future Research Directions

To continue advancing the field of autonomous self-healing infrastructure, we identify several key research avenues:

1. **Trust-Aware Reinforcement Learning:** Future work should integrate uncertainty quantification techniques—such as Bayesian neural networks or ensemble models—into RL agents to estimate the confidence of each recovery action. When the agent's confidence falls below a defined threshold, the system would escalate the decision to human operators or invoke additional diagnostic routines, ensuring safe handling of critical scenarios like security breaches or data corruption. Research should explore efficient methods for computing uncertainty in real time without compromising performance.
2. **Policy-as-Code Compliance Automation:** Embedding regulatory requirements directly into remediation playbooks will be crucial for industries subject to strict data governance (e.g., GDPR, HIPAA, PCI-DSS). Future frameworks should implement policy-as-code engines that validate recovery actions against compliance rules before execution, automatically generating audit trails and ensuring evidence of adherence. Investigations should focus on standardizing compliance policies, integrating policy-verification checks into IaC pipelines, and measuring the impact of compliance enforcement on recovery latency.
3. **Federated Learning for Distributed Resilience:** As organizations deploy workloads across multiple geographic regions or hybrid cloud environments, sharing learning models without centralized data pooling becomes imperative for privacy and data sovereignty. Research should develop federated learning protocols tailored to self-healing, enabling local clusters to train anomaly detectors and RL policies on-site, then securely aggregate model updates. Challenges include designing communication-efficient aggregation methods, handling non-IID data distributions, and ensuring robustness against adversarial updates.
4. **Edge and IoT Adaptation:** Extending self-healing capabilities to edge computing and IoT devices demands lightweight, resource-efficient agents. Future studies should explore model compression, quantization, and on-device learning techniques to adapt RL and anomaly detection algorithms for constrained hardware. Additionally, research must address intermittent connectivity, decentralized orchestration, and cross-layer coordination between edge nodes and central cloud controllers to maintain consistent resilience.
5. **Explainable AI for Auditability:** For critical applications—such as healthcare, finance, and autonomous vehicles—understanding the rationale behind automated recovery decisions is essential. Future research should focus on integrating explainable AI techniques (e.g., attention mechanisms, saliency maps, or counterfactual explanations) into RL agents and anomaly detectors. By providing interpretable justifications

for selected actions, these methods will enhance operator trust, enable regulatory compliance, and facilitate debugging of self-healing processes.

8. Conclusion

In this study, we presented a comprehensive, multi-layered self-healing framework designed to autonomously detect, diagnose, and remediate failures within cloud infrastructures. By synergizing predictive analytics for proactive anomaly detection, reinforcement learning (RL) for dynamic recovery optimization, and rule-based automation for immediate playbook-driven responses, the framework addresses the full spectrum of failure scenarios—from predictable faults to novel, complex incidents.

Our empirical evaluation across AWS and Azure Kubernetes clusters, encompassing over 220 systematically injected fault scenarios, demonstrated impactful improvements: MTTR was reduced by 85% (from 90 minutes to 13.5 minutes), recovery reliability exceeded 95%, fault tolerance reached over 91%, and overall system uptime surpassed 98% during a continuous 30-day period. Importantly, these benefits were achieved with a modest computational overhead of under 10%, affirming the framework's operational efficiency.

The integration of Infrastructure as Code (IaC) practices—leveraging Terraform, Helm, and GitOps—ensures consistent and secure multi-cloud deployments, while Vault-managed secrets and Argo CD–driven rollbacks enhance production readiness and compliance. Sensitivity analyses further confirmed the framework's robustness to threshold calibrations, maintaining MTTR improvements within ±5% under varied detection settings.

Beyond quantitative gains, our framework bridges the gap between reactive, isolated remediation methods and fully autonomous recovery systems by providing a unified, extensible architecture. This enables organizations to reduce human intervention, accelerate incident resolution, and improve service-level adherence while maintaining flexibility to adapt to evolving infrastructure landscapes.

In conclusion, this work offers a practical blueprint for next-generation cloud resilience, combining advanced AI techniques with robust engineering practices. The proposed framework not only elevates operational reliability but also lays the groundwork for future innovations in trust-aware automation, compliance-driven recovery, and edge-native self-healing solutions. We invite practitioners and researchers to adopt and extend this model to realize truly autonomous and resilient cloud ecosystems.

References

[1] Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36(1), 41–50. https://doi.org/10.1109/MC.2003.1160055.
[2] Gartner. (2023). Cloud Downtime and Resilience Report. *Gartner Insights*, 19(3). https://www.gartner.com/
[3] IDC. (2023). Global Cloud Survey. *IDC Research Reports*, Q4 2023. https://www.idc.com/
[4] Jiang, X., Li, Y., & Liu, Q. (2021). Anomaly Detection in Cloud Environments: A Survey. *Journal of Cloud Computing*, 10(4), 205–222. https://doi.org/10.1186/s13677-021-00254-3.
[5] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press. https://mitpress.mit.edu/books/reinforcement-learning-second-edition
[6] Abadi, M., et al. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. *arXiv preprint arXiv:1603.04467*. https://arxiv.org/abs/1603.04467
[7] Williams, R. J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3–4), 229–256. https://doi.org/10.1007/BF00992696.
[8] Dean, J., et al. (2012). Large Scale Distributed Deep Networks. *Advances in Neural Information Processing Systems*, 25, 1223–1231. https://papers.nips.cc/paper/2012/hash/6aca97005c68f1206823815f66102863-Abstract.html.
[9] Ousterhout, J. K. (2013). Why Data Centers Need Kernel Support for Multi-Core Tsunami. *Communications of the ACM*, 56(7), 44–48. https://doi.org/10.1145/2483852.2483865
[10] Kreps, J., Narkhede, N., & Rao, J. (2011). Kafka: A Distributed Messaging System for Log Processing. *Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB)*. https://researcher.watson.ibm.com/researcher/files/us-kreps/netdb11.pdf