JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

KAIRO: A SELF-HEALING DEPLOYMENT FRAMEWORK USING CONFIDENCE-BASED DECISION MODELS AND RETRIEVAL-AUGMENTED KNOWLEDGE MEMORY

Kunal Chikte
Senior Software Engineer Pune, India

Abstract: *Modern deployment environments require engineers to manually resolve configuration issues, dependency conflicts, and infrastructure failures — a process that is slow, costly, and error-prone. Existing CI/CD pipelines for automate execution but offer no autonomous recovery or learning capabilities. This paper proposes Kairo, a Self-Healing Deployment Framework that integrates confidence-based decision making, retrieval-augmented knowledge storage, internet-assisted problem resolution, and vector-based memory systems. Kairo continuously learns from deployment failures by discovering solutions from external sources, evaluating solution confidence, storing validated resolutions in a vector database, and reusing successful remediation strategies for future incidents. The framework reduces mean-time-to-recovery (MTTR), minimizes human intervention, and improves deployment reliability through adaptive, experience-driven learning.*

Index Terms *- Self-Healing Systems, DevOps, Deployment Automation, Retrieval-Augmented Generation, Vector Database, Autonomous Remediation, Artificial Intelligence*

I. INTRODUCTION

Software deployment remains one of the most demanding phases of the software development lifecycle. Although Continuous Integration and Continuous Deployment (CI/CD) pipelines have streamlined build and release processes, failure handling continues to depend on experienced engineers who manually investigate logs, identify root causes, and apply fixes — often under production pressure.

Common deployment failures include missing dependencies, configuration mismatches, container runtime errors, infrastructure resource constraints, network configuration issues, and security policy violations. In each case, traditional systems terminate execution, generate logs, and wait for human intervention. This reactive approach is expensive and does not improve over time.

Kairo addresses these limitations by introducing an autonomous deployment framework capable of detecting failures, retrieving historically validated solutions, evaluating remediation confidence, and conducting external research when no known solution is available. By persisting validated solutions in a vector database, the framework continuously learns from operational experience, reducing repeated manual effort, and improving system reliability.

II. PROBLEM STATEMENT

Current deployment automation systems suffer from the following structural limitations:

- No adaptive learning — resolved incidents are not retained for future use.
- Repeated investigation — identical failures require fresh human analysis each time.
- Expert dependency — resolution quality is contingent on available engineer expertise.
- High incident resolution time — MTTR remains elevated due to manual workflows.
- Absence of persistent operational knowledge — remediation insights are lost when engineers rotate or leave.

The cumulative effect is a deployment ecosystem that is brittle by design: it automates execution but externalizes failure recovery entirely to human operators.

III. PROPOSED FRAMEWORK

Kairo is structured around four integrated modules that collectively enable autonomous failure recovery and continuous learning.

A. Deployment Knowledge Repository

A structured store containing deployment workflows, infrastructure procedures, operational guidelines, and a history of past deployment incidents with their resolutions. This repository is the primary knowledge substrate for the decision engine.

B. Confidence-Based Decision Engine

The core reasoning component evaluates candidate solutions by computing a composite confidence score derived from historical success rate, semantic similarity to the current failure context, and prior validation accuracy. Solutions meeting a configurable threshold are applied autonomously; those falling short trigger the Autonomous Research Engine.

C. Autonomous Research Engine

When no sufficiently confident solution exists in the knowledge repository, Kairo consults trusted external technical sources — documentation, security advisories, and community knowledge bases — to discover candidate remediation strategies. Discovered solutions are evaluated before being applied and stored.

D. Self-Learning Memory Layer

Validated solutions are embedded using a language model and stored in a pgVector-backed vector database. Future incident queries retrieve semantically similar past resolutions, enabling the system to recognize and respond to recurring failure patterns with increasing precision.

IV. MATHEMATICAL MODEL

Let E denote a deployment error, S a candidate solution, and C the confidence score assigned to S given E. The confidence function is defined as:

$$C(S, E) = \alpha \cdot H(S) + \beta \cdot SM(S, E) + \gamma \cdot V(S)$$

Where:

- H(S) = Historical success rate of solution S across prior incidents
- SM (S, E) = Semantic similarity score between S and the current error context E
- V(S) = Validation accuracy of S from prior deployment cycles
- α, β, γ are weighting coefficients such that $\alpha + \beta + \gamma = 1$ The decision rule applied by the framework is:

$$\text{If } C(S, E) \geq T \rightarrow \text{Apply Solution } S$$

$$\text{If } C(S, E) < T \rightarrow \text{Initiate Autonomous Research}$$

The threshold T is configurable per deployment environment and failure criticality class. Confidence values are updated after every deployment cycle, enabling weights to evolve with operational experience.

V. METHODOLOGY

Kairo operates through a structured nine-phase execution pipeline activated upon each deployment failure:

- Phase 1: Deployment execution is initiated through the CI/CD pipeline.
- Phase 2: The error detection module captures failure logs and classifies the incident type.
- Phase 3: A vector similarity search retrieves semantically related previously resolved incidents.
- Phase 4: The confidence engine scores each candidate's solution and ranks them.
- Phase 5: The highest-confidence solution above threshold T is applied autonomously.
- Phase 6: Automated validation tests verify that the remediation was successful.
- Phase 7: If validation fails, the Autonomous Research Engine is activated.
- Phase 8: Discovered solutions are evaluated, scored, and — if validated — applied.
- Phase 9: The knowledge repository and vector database are updated with the new validated resolution.

VI. SYSTEM ARCHITECTURE

Fig. 1 illustrates the complete system architecture of Kairo. The framework operates as a closed feedback loop: deployment failures trigger error detection, which feeds into vector retrieval and confidence scoring. The decision module either applies to a known solution or delegates to the research engine. All validated resolutions persisted in the vector database to continuously enrich the knowledge layer.

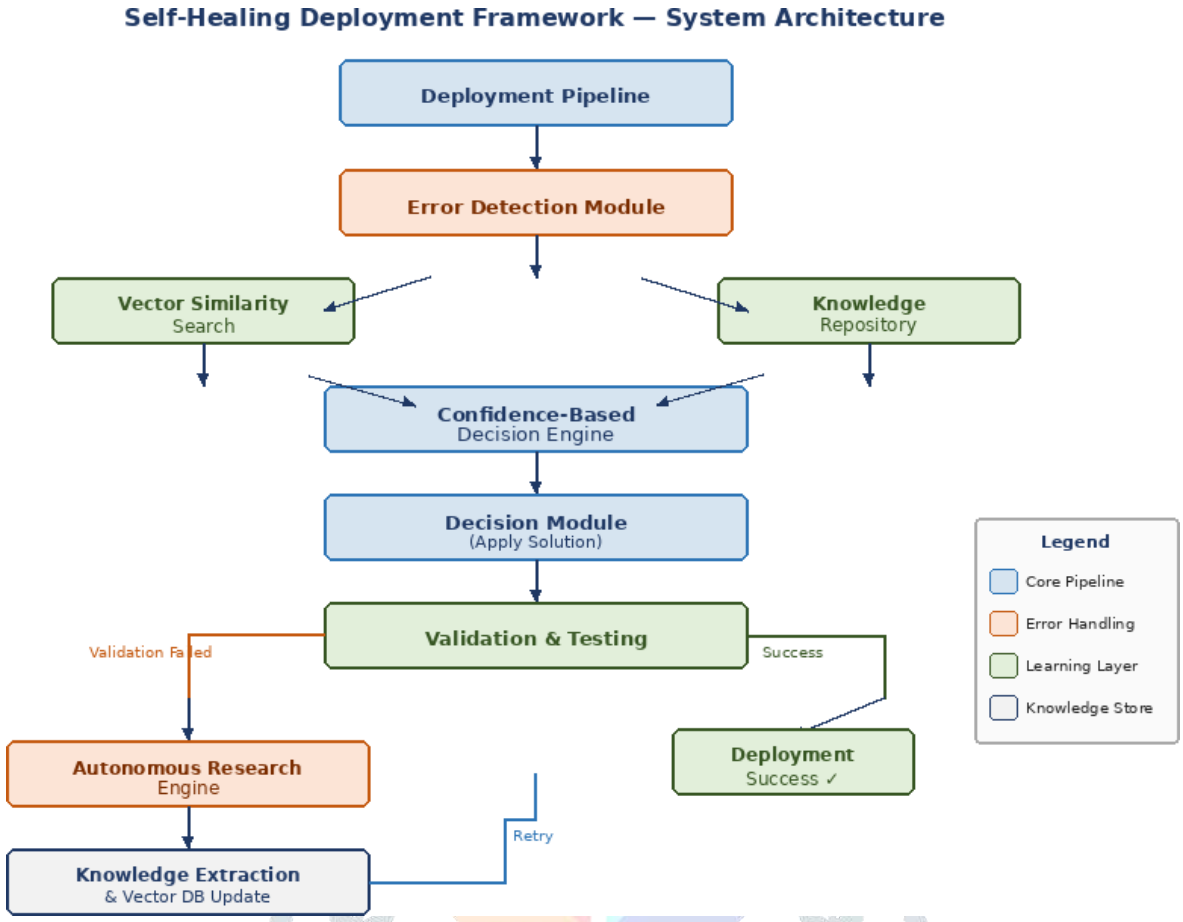

Figure 1: Kairo Self-Healing Deployment Framework — System Architecture

The architecture is technology-agnostic at the orchestration level and is implemented using Large Language Models for semantic understanding, pgVector for embedding storage and retrieval, Docker for containerized deployment environments, and cloud infrastructure APIs for resource management.

VII. IMPLEMENTATION

The prototype implementation of Kairo utilizes the following technology stack:

- Large Language Models (LLMs): Semantic analysis of error logs, solution ranking, and research query synthesis.
- pgVector (PostgreSQL): Efficient cosine-similarity search over solution embeddings.
- Embedding Models: Sentence-level embeddings convert error contexts and solutions into dense vector representations.
- Retrieval-Augmented Generation (RAG): Structured retrieval combined with generative reasoning to synthesize candidate solutions.
- Docker: Containerized deployment environments enabling reproducible failure of simulation and validation.
- Cloud Infrastructure APIs: Programmatic access to compute, networking, and storage for autonomous remediation.

VIII. ADVANTAGES

Kairo offers measurable improvements over conventional deployment automation:

- Reduced deployment downtime through faster, autonomous failure resolution.
- Lower operational costs by reducing reliance on on-call engineering availability.
- Continuous learning capability that improves resolution accuracy with each incident.
- Improved deployment success rate as the knowledge base matures over time.
- Reduced expert dependency, enabling smaller teams to maintain complex deployment environments.
- Faster incident recovery, with MTTR decreasing confidence thresholds are met by accumulated data.

IX. FUTURE WORK

Planned extensions to the Kairo framework include:

- Multi-cloud deployment support with provider-aware solution routing.
- Reinforcement learning optimization to dynamically adjust confidence weights α, β, and γ without manual reconfiguration.
- Predictive failure prevention using time-series analysis of deployment telemetry.
- Security-aware autonomous remediation with integrated policy enforcement checks.

- Enterprise-scale distributed memory supporting multi-team and multi-region knowledge federation.

X. CONCLUSION

This paper presented Kairo, a self-healing deployment framework designed to transform static automation pipelines into adaptive, experienced-driven systems. By combining confidence-based decision making with retrieval-augmented knowledge memory and autonomous research capabilities, Kairo enables deployment infrastructure to detect failures, retrieve validated solutions, conduct external research when required, and continuously expand its operational knowledge base.

The framework addresses a fundamental gap in current DevOps tooling: the inability to learn from past incidents. As deployment environments grow in complexity and scale, the capacity for autonomous, intelligent failure recovery will be essential. Kairo provides a principled and extensible foundation for this capability.

. REFERENCES

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, Special Publication 800-145, 2011.

[2] G. Kim, J. Humble, P. Debois, and J. Willis, The DevOps Handbook. Portland, OR: IT Revolution Press, 2016.

[3] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems (NeurIPS), 2020.

[4] J. Johnson, M. Douze, and H. Jégou, "Billion-Scale Similarity Search with GPUs," IEEE Transactions on Big Data, 2019.

[5] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," IEEE Computer, vol. 36, no. 1, pp. 41–50, 2003.

[6] J. Humble and D. Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Boston, MA: Addison-Wesley, 2010.

[7] J. Cito, P. Leitner, T. Fritz, and H. C. Gall, "The Making of Cloud Applications: An Empirical Study on Software Development for the Cloud," Proc. ACM SIGSOFT Symp. Foundations of Software Engineering, 2015.

[8] A. Vaswani et al., "Attention is All You Need," Advances in Neural Information Processing Systems (NeurIPS), 2017.