Ripusoodan Sharma 2026, 14:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

ARCH: An LLM-Driven Autonomous Self-Healing Framework for Cloud Operations Using Retrieval-Augmented Generation

Ripusoodan Sharma, Dr. Kirti Jain
Sanjeev Agrawal Global Educational University Bhopal

Abstract- The increasing complexity of cloud-native environments has made fault detection and recovery a critical challenge for modern IT operations. Traditional AIOps solutions rely heavily on static rules or data-driven models, which often lack adaptability in dynamic and unpredictable scenarios. To address these limitations, this paper proposes the ARCH (Autonomous Reasoning and Contextual Healing) framework, an intelligent self-healing system that integrates Large Language Models (LLMs) with Retrieval-Augmented Generation (RAG) for autonomous cloud operations. The proposed framework follows a layered architecture consisting of perception, cognition, knowledge, and action components, enabling continuous monitoring, contextual reasoning, and automated remediation. By leveraging advanced reasoning strategies such as chain-of-thought and action-oriented decision-making, the system dynamically analyzes telemetry data and executes corrective actions with minimal human intervention [5], [7]. Furthermore, the incorporation of RAG enhances the system's ability to utilize historical incident data, thereby improving decision accuracy and contextual awareness [4]. The effectiveness of the ARCH framework is evaluated using key performance metrics, including Mean Time to Repair (MTTR), Autonomous Success Rate (ASR), and overall system efficiency. Experimental results demonstrate that the proposed approach significantly improves fault resolution performance, achieving up to 82% reduction in MTTR and 89.5% autonomous success rate compared to conventional methods. These findings highlight the potential of LLM-driven architectures in enabling scalable and intelligent self-healing cloud systems.

Keywords—AIOps, Large Language Models (LLMs), Autonomous Systems, Self-Healing, Cloud Computing, Retrieval-Augmented Generation (RAG), Generative AI, Intelligent Automation.

I. INTRODUCTION

Cloud computing has emerged as a fundamental platform for delivering scalable and resilient digital services across diverse application domains. However, the rapid growth of cloud-native architectures, microservices, and distributed systems has significantly increased operational complexity. As a result, fault detection, diagnosis, and recovery have become critical challenges for modern IT operations. Traditional approaches, including rule-based monitoring and conventional machine learning techniques, often struggle to adapt to dynamic environments and unforeseen failure patterns [10], [13].

In recent years, Artificial Intelligence for IT Operations (AIOps) has been introduced to automate system monitoring and anomaly detection. While AIOps solutions improve operational efficiency, they are typically limited by their reliance on predefined rules or static models, which restrict their ability to handle complex and context-dependent failures [12], [13]. Consequently, there is a growing need for more intelligent and adaptive systems capable of understanding system behavior and autonomously performing corrective actions.

The emergence of Large Language Models (LLMs) has opened new opportunities for intelligent automation by enabling advanced reasoning and contextual understanding. Techniques such as

Chain-of-Thought (CoT) prompting and ReAct (Reasoning and Acting) frameworks allow models to perform step-by-step reasoning and decision-making in complex scenarios [5], [7]. Furthermore, Retrieval-Augmented Generation (RAG) enhances LLM capabilities by integrating external knowledge sources, enabling more accurate and context-aware responses [4].

Motivated by these advancements, this paper proposes the ARCH (Autonomous Reasoning and Contextual Healing) framework, a novel approach for enabling self-healing cloud systems. The framework integrates LLM-driven reasoning with RAG-based knowledge retrieval in a layered architecture consisting of perception, cognition, knowledge, and action components. This design enables continuous monitoring, intelligent analysis, and automated remediation through a closed-loop feedback mechanism.

The main contributions of this paper are summarized as follows:

- A novel layered architecture for autonomous cloud self-healing integrating LLMs and RAG.
- An agent-based reasoning framework leveraging CoT and ReAct strategies for adaptive decision-making.
- A closed-loop self-healing mechanism for continuous fault detection and remediation.
- Comprehensive experimental evaluation demonstrating significant improvements in MTTR, ASR, and system efficiency.

The remainder of this paper is organized as follows. Section 2 reviews related work in AIOps and LLM-based systems. Section 3 presents the proposed ARCH architecture. Section 4 describes the methodology and implementation details. Section 5 discusses experimental results and performance evaluation. Section 6 outlines limitations and future directions, followed by the conclusion in Section 7.

II. LITERATURE REVIEW

The increasing complexity of cloud computing environments has driven significant research in the field of Artificial Intelligence for IT Operations (AIOps). Early approaches primarily focused on rule-based systems and traditional machine learning techniques for anomaly detection and fault diagnosis. These methods, although effective in controlled environments, often lack adaptability and struggle to generalize across dynamic and large-scale cloud infrastructures [12], [13].

Recent advancements have explored the use of deep learning techniques for log analysis, anomaly detection, and predictive maintenance in cloud systems. For instance, log-based anomaly detection models have demonstrated improved accuracy in identifying system faults; however, they still rely heavily on historical data and lack contextual reasoning capabilities [14]. Similarly, reinforcement learning-based approaches have been proposed for autonomous cloud management, enabling systems to learn optimal recovery strategies over time. Despite their potential, such methods require extensive training and are often computationally expensive [12].

The emergence of Large Language Models (LLMs) has introduced a new paradigm for intelligent automation. Techniques such as Chain-of-Thought (CoT) prompting enable models to perform structured reasoning, improving their ability to solve complex tasks [5]. Additionally, the ReAct framework combines reasoning with action execution, allowing models to interact with external systems and make dynamic decisions [7]. These capabilities make LLMs particularly suitable for tasks involving diagnosis and remediation in cloud environments.

To further enhance the reasoning capabilities of LLMs, Retrieval-Augmented Generation (RAG) has been proposed as an effective approach for integrating external knowledge sources. RAG-based systems retrieve relevant contextual information from historical data or knowledge bases, enabling more accurate and informed decision-making [4]. This approach has been successfully applied in knowledge-intensive tasks and is increasingly being adopted in autonomous agent systems.

Recent research has also focused on the development of multi-agent frameworks for

autonomous cloud operations. Such systems leverage multiple intelligent agents to collaboratively monitor, analyze, and manage cloud environments [2], [3]. These frameworks demonstrate promising results in improving system resilience and reducing human intervention. However, challenges such as coordination complexity, latency, and scalability remain open research problems.

Despite these advancements, existing solutions often lack a unified framework that integrates perception, reasoning, knowledge retrieval, and action execution in a cohesive manner. Most approaches address individual components of the self-healing process but fail to provide an end-to-end autonomous system.

To address these gaps, this paper proposes the ARCH framework, which combines LLM-based reasoning, RAG-driven knowledge retrieval, and a layered architecture to enable fully autonomous and adaptive self-healing in cloud environments.

III. PROPOSED ARCH ARCHITECTURE

This section presents the proposed ARCH (Autonomous Reasoning and Contextual Healing) framework, designed to enable intelligent and autonomous self-healing in cloud environments. The architecture integrates Large Language Models (LLMs) with Retrieval-Augmented Generation (RAG) to provide contextual reasoning and adaptive fault remediation. The overall system is organized into a layered structure that facilitates modularity, scalability, and continuous feedback-driven improvement.

The overall architecture of the proposed ARCH framework is illustrated in Fig. 1, which demonstrates the interaction between perception, cognition, knowledge, and action layers in a closed-loop self-healing system.

Architecture Overview

The ARCH framework follows a four-layered design consisting of perception, cognition, knowledge, and action components. Each layer performs a distinct function while interacting with others to enable end-to-end autonomous operation.

The perception layer is responsible for collecting telemetry data, including logs, metrics, and traces, from cloud infrastructure. This data forms the foundation for anomaly detection and system monitoring.

The cognition layer acts as the reasoning engine of the framework. It leverages LLM-based techniques to analyze incoming data, generate hypotheses, and determine appropriate remediation actions. Advanced reasoning strategies such as Chain-of-Thought and ReAct enable the system to handle complex and multi-step decision-making processes.
The knowledge layer integrates Retrieval-Augmented Generation (RAG) to enhance contextual awareness. It retrieves relevant historical incidents, documentation, and prior solutions from a knowledge base, allowing the system to make informed decisions based on past experiences.
The action layer is responsible for executing remediation steps through cloud APIs and orchestration tools. It ensures that corrective actions are applied safely and effectively to restore system stability.

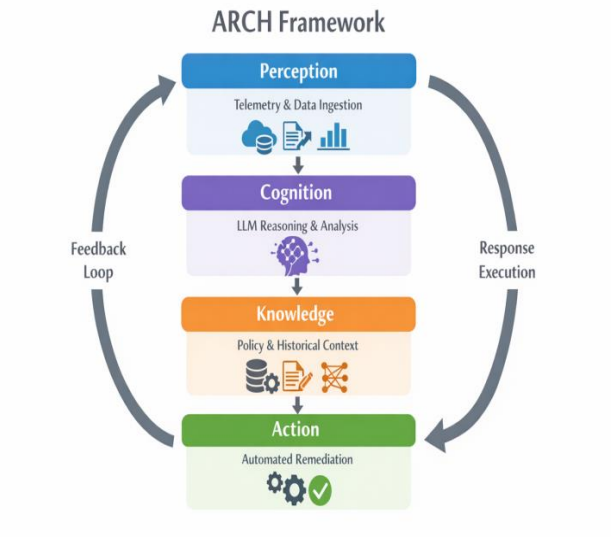

Figure. 1. ARCH Framework: Layered architecture showing Perception, Cognition, Knowledge, and Action with a continuous self-healing feedback loop.

Layered Functional Breakdown

- **Perception Layer**

The perception layer continuously monitors the cloud environment by collecting telemetry data from various sources such as system logs, performance metrics, and distributed traces. This layer plays a critical role in identifying anomalies and triggering the self-healing process. Efficient data ingestion and preprocessing mechanisms ensure timely detection of potential failures.

- **Cognition Layer (The Reasoning Brain)**

The cognition layer forms the core intelligence of the ARCH framework. It utilizes LLMs to interpret telemetry data, generate reasoning steps, and propose corrective actions. The internal reasoning mechanism of the agent is illustrated in Fig. 3, which follows an iterative cycle of thought, action, observation, and refined thought to enable adaptive decision-making.

By incorporating structured reasoning techniques, the system can analyze complex failure scenarios and dynamically adjust its decisions. This iterative reasoning process enhances accuracy and reduces the likelihood of incorrect or repetitive actions.

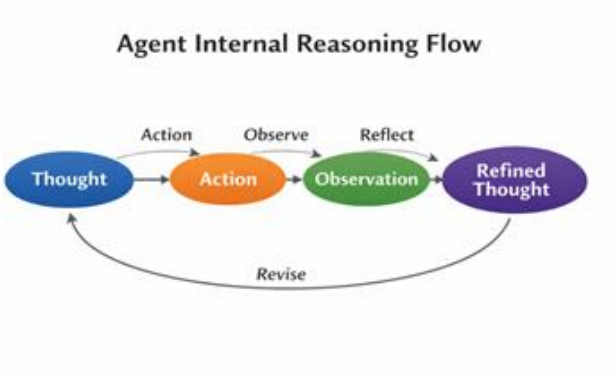

Figure. 3. Agent Internal Reasoning Flow: Iterative process of thought generation, action execution, observation, and refined reasoning for autonomous decision-making.

Knowledge Layer (RAG Integration)

The knowledge layer enhances the reasoning capability of the system by incorporating Retrieval-Augmented Generation. It retrieves relevant contextual information from historical logs, incident databases, and documentation repositories. This enables the system to leverage prior knowledge and improve the quality of its decisions.

The integration of RAG ensures that the reasoning process is grounded in factual and contextually relevant data, thereby reducing hallucinations and improving reliability.

- **Action Layer**

The action layer is responsible for executing the decisions generated by the cognition layer. It interacts with cloud management systems and APIs to perform remediation tasks such as restarting services, scaling resources, or updating configurations.

To ensure safety and reliability, the framework incorporates validation mechanisms before executing any action. These guardrails prevent unintended or harmful operations, thereby maintaining system integrity.

The Agentic Self-Healing Loop

The ARCH framework operates as a continuous feedback-driven system, enabling iterative improvement and adaptive behavior. The complete self-healing lifecycle is illustrated in Fig. 2, showing a circular workflow from telemetry ingestion to autonomous execution.

The process begins with data collection in the perception layer, followed by contextual reasoning in the cognition layer. The knowledge layer provides relevant information to support decision-making, and the action layer executes remediation steps. The feedback loop ensures that the system continuously evaluates the outcome of its actions and refines its future decisions.

This closed-loop design enables the ARCH framework to achieve efficient, scalable, and autonomous self-healing in complex cloud environments.

Figure. 2. Self-Healing Loop: Circular workflow illustrating telemetry ingestion

IV. Methodology

This section describes the experimental setup, implementation strategy, and evaluation methodology used to validate the proposed ARCH framework. The objective is to assess the effectiveness of the system in detecting faults, performing reasoning, and executing automated remediation in cloud environments.

Experimental Setup

The experimental environment consists of a cloud-based infrastructure with simulated microservices and distributed workloads. Telemetry data, including logs, metrics, and traces, are continuously collected to emulate real-world operational conditions. Fault scenarios such as service crashes, resource exhaustion, and configuration errors are intentionally injected to evaluate the robustness of the system.

The ARCH framework is deployed as an agent-based system, where each component interacts with cloud services through APIs. The Large Language Model (LLM) is used for reasoning, while the Retrieval-Augmented Generation (RAG) module retrieves relevant contextual information from historical incident data and knowledge repositories.

Fault Injection Strategy

To evaluate system performance under realistic conditions, various fault scenarios are introduced into the environment. These include application-level failures, infrastructure-level disruptions, and performance degradation issues. The injected faults are designed to test the system's ability to detect anomalies, analyze root causes, and apply appropriate remediation actions.

Each fault scenario is monitored using telemetry data, and the system's response is recorded for further analysis. This approach ensures a comprehensive evaluation of the framework across different types of failures.

ARCH Agentic Implementation

The ARCH framework is implemented as an autonomous agent that follows a structured reasoning and execution process. The overall working mechanism is summarized in Algorithm 1.

Algorithm 1: ARCH Self-Healing Process

Input: Telemetry data (logs, metrics, traces)
Output: Autonomous fault detection and remediation

- Collect telemetry data from cloud environment
- Retrieve relevant historical incidents using RAG
- Generate reasoning using LLM (CoT / ReAct)
- Validate proposed action using safety guardrails
- Execute remediation action via cloud APIs
- Verify system state after execution
- If fault persists, update context and repeat from Step 2
- Else, log successful remediation and update knowledge base

The algorithm highlights the closed-loop interaction between perception, reasoning, and execution components. It ensures continuous monitoring and adaptive remediation through iterative feedback.

Performance Metrics

To quantitatively evaluate the performance of the ARCH framework, multiple metrics are considered.

Mean Time to Repair (MTTR): This metric represents the average time required to recover from failures.

$$(1)\ MTTR = \frac{Total\ Repari\ Time}{Number\ of\ Incidents}$$

This metric evaluates the efficiency of the system in minimizing downtime.

Reasoning Latency: The total time taken by the system to detect, analyze, and resolve a fault.

$$Total\ Latency = Detection + Reasoning + Execution$$

This metric captures the delay introduced during detection, reasoning, and execution phases.

Self-Healing Efficiency (η): This metric measures the effectiveness of automated recovery compared to manual intervention.

$$(2)\ n = \frac{\sum_{i=1}^{n}(T_{manual,i} - T_{auto,i})}{n.T_{manual}}$$

A higher value of η indicates better system performance in reducing recovery time and improving operational efficiency.

The combination of these metrics provides a comprehensive evaluation of system performance, covering both accuracy and efficiency aspects.

V. RESULTS AND DISCUSSION

This section presents the experimental results obtained from the evaluation of the proposed ARCH framework. The performance of the system is analyzed using quantitative metrics, including Mean Time to Repair (MTTR), Autonomous Success Rate (ASR), precision, recall, and overall efficiency. The results are compared with baseline approaches such as traditional AIOps and deep learning-based methods.

Quantitative Performance Analysis

To evaluate the detection performance of the proposed system, standard metrics such as precision and recall are considered.

$$Precsision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

These metrics provide insights into the accuracy and completeness of fault detection. High precision indicates fewer false positives, while high recall reflects the system's ability to detect most of the actual faults.

The comparative performance of the ARCH framework against baseline models is summarized in Table II. The results demonstrate significant improvements in both detection accuracy and recovery efficiency.

As shown in Fig. 4, the ARCH framework significantly reduces the Mean Time to Repair (MTTR) compared to traditional approaches. The reduction in MTTR highlights the effectiveness of automated reasoning and rapid remediation capabilities.

The comparative performance of the ARCH framework against baseline models is presented in Table II.

Table II. Performance Comparison of AIOps, Deep Learning, and ARCH Framework

Metric	Traditional AIOps (Static Rules)	Deep Learning (LSTM/CNN)	ARCH (Agentic GenAI)	Improvement (vs. DL)
Detection Precision	68.4%	84.2%	96.1%	+11.9%
MTTR (Minutes)	24.5	18.2	4.2	-76.9%
ASR (%)	12.0%	45.0%	89.5%	+44.5%

Metric	Traditional AIOps (Static Rules)	Deep Learning (LSTM/CNN)	ARCH (Agentic GenAI)	Improvement (vs. DL)
False Positive Rate	15.2%	8.4%	2.1%	-6.3%

The results indicate that the ARCH framework outperforms baseline approaches in terms of both detection accuracy and recovery efficiency. A significant reduction in MTTR and improvement in autonomous success rate can be observed.

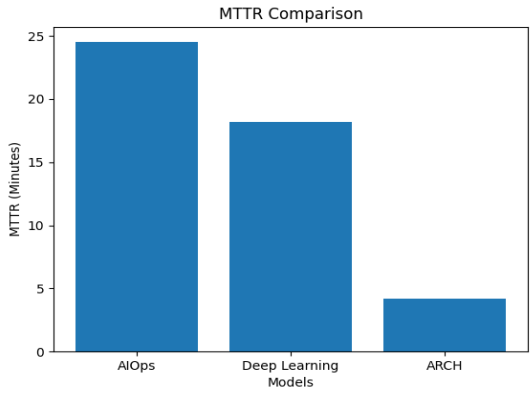

Figure. 4. MTTR Comparison across AIOps, Deep Learning, and ARCH framework.

In addition to MTTR, the Autonomous Success Rate (ASR) is evaluated to measure the system's ability to resolve faults without human intervention. The ARCH framework achieves a substantially higher success rate, as illustrated in Fig. 5.

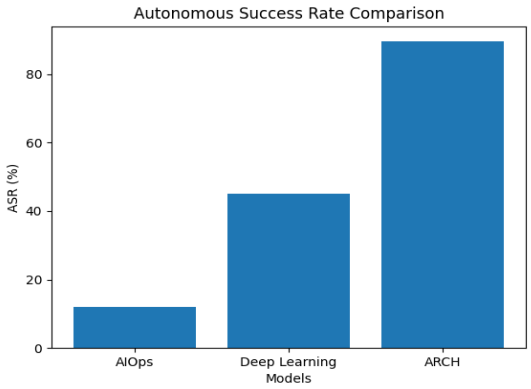

Figure. 5. Autonomous Success Rate (ASR) comparison across different approaches.

The improvements observed in MTTR and ASR can be attributed to the integration of LLM-based reasoning and RAG-driven contextual understanding, which enable more accurate and efficient decision-making.

Discussion

The experimental results clearly indicate that the ARCH framework outperforms conventional AIOps and deep learning-based solutions across multiple metrics. The use of structured reasoning techniques allows the system to analyze complex fault scenarios more effectively, while the knowledge retrieval mechanism enhances contextual awareness.

One of the key advantages of the proposed approach is its ability to adapt to dynamic environments. Unlike traditional systems that rely on predefined rules, the ARCH framework continuously refines its decisions through feedback loops, leading to improved performance over time.

However, the system also introduces certain trade-offs. The incorporation of LLM-based reasoning increases computational overhead and may introduce latency in time-sensitive scenarios. Despite these challenges, the overall benefits in terms of accuracy, automation, and efficiency outweigh the limitations.

Efficiency of Self-Healing

To further evaluate system performance, the self-healing efficiency (η) is analyzed. This metric

captures the relative improvement in recovery time achieved through automation.

As depicted in Fig. 6, the ARCH framework demonstrates significantly higher efficiency compared to baseline approaches. The results indicate that automated remediation not only reduces downtime but also improves overall system reliability.

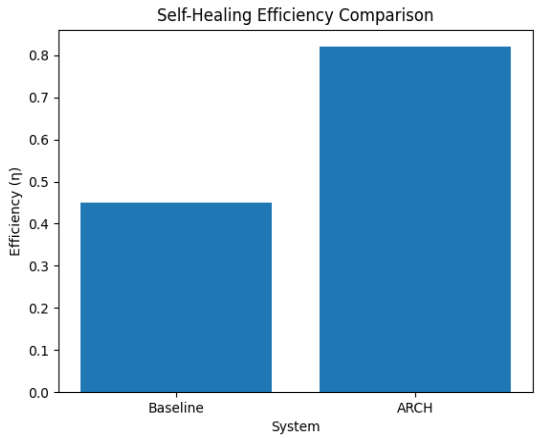

Figure. 6. Self-Healing Efficiency (η) comparison between baseline and ARCH framework.

The consistent improvement across all evaluation metrics validates the effectiveness of the proposed architecture and highlights its potential for real-world deployment in cloud environments.

VI. CHALLENGES AND FUTURE SCOPE

While the proposed ARCH framework demonstrates significant improvements in autonomous cloud self-healing, several challenges remain that must be addressed to enable large-scale real-world deployment.

Current Challenges

Despite the advantages of integrating Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG), the system faces certain technical and practical limitations.

- **Hallucination in LLMs**

LLMs may occasionally generate incorrect or misleading outputs, commonly referred to as hallucinations. In the context of cloud operations, such inaccuracies can lead to inappropriate remediation actions, potentially affecting system stability. Although RAG helps reduce this issue by grounding responses in relevant data, complete elimination of hallucination remains a challenge.

- **Latency Overhead**

The reasoning process in LLM-based systems introduces additional latency compared to traditional rule-based approaches. The time required for context retrieval, reasoning, and response generation may impact performance in time-critical scenarios.

- **Context Window Limitations**

LLMs operate within a fixed context window, which restricts the amount of information that can be processed at a given time. In large-scale cloud systems with extensive logs and telemetry data, this limitation may affect the completeness of analysis.

- **Computational Cost**

The deployment of large-scale LLMs involves significant computational and financial overhead. Continuous processing of telemetry data, frequent reasoning cycles, and interaction with external APIs result in increased resource consumption. This may limit the feasibility of deploying such systems in cost-sensitive environments.

- **Security and Trust Concerns**

The integration of autonomous agents with cloud control mechanisms introduces potential security risks. Incorrect or malicious actions could impact critical system components. Additionally, the use of external LLM services may raise concerns related to data privacy, confidentiality, and regulatory compliance.

- **Real-Time Constraints**

Although the ARCH framework improves fault resolution efficiency, the inherent latency of LLM-based reasoning may limit its applicability in real-time or latency-sensitive applications. Systems requiring immediate response may not fully benefit from the current implementation.

- **Future Scope**

Future research can address these challenges and further enhance the capabilities of the ARCH framework.

First, the development of lightweight and optimized LLMs can help reduce computational overhead and improve scalability. Techniques such as model compression and on-device inference may enable cost-effective deployment.

Second, incorporating advanced guardrail mechanisms and validation strategies can improve system safety and reliability. This includes integrating rule-based constraints and anomaly verification techniques before executing actions.

Third, hybrid architectures combining LLMs with traditional AIOps techniques can provide a balance between efficiency and accuracy, particularly in latency-sensitive environments.

Fourth, expanding the knowledge base and improving retrieval strategies in RAG can enhance contextual understanding and reduce dependency on model-generated reasoning.

Finally, real-world deployment and large-scale validation in production cloud environments will be essential to evaluate the robustness and practical applicability of the proposed system.

REFERENCES

1. A.-M. Tanasă, S.-V. Oprea, and A. Bâra, "Designing an Architecture of a Multi-Agentic AI-Powered Virtual Assistant Using LLMs and RAG," Electronics, vol. 15, no. 2, p. 334, 2026.
2. K. Parthasarathy et al., "Engineering LLM Powered Multi-agent Framework for Autonomous CloudOps," arXiv:2501.08243, 2025.
3. Y. Chen et al., "AIOPSLAB: A Holistic Framework to Evaluate AI Agents for Enabling Autonomous Clouds," Microsoft Research, 2024.
4. H. Liu et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Proc. NeurIPS, 2024.
5. J. Wei et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," arXiv:2201.11903, 2022.
6. T. Schick et al., "Toolformer: Language Models Can Teach Themselves to Use Tools," arXiv:2302.04761, 2023.
7. S. Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," arXiv:2210.03629, 2023.
8. OpenAI, "GPT-4 Technical Report," arXiv:2303.08774, 2023.
9. M. Chen et al., "Evaluating Large Language Models Trained on Code," arXiv:2107.03374, 2021.
10. S. Zhang et al., "A Survey on AIOps: Intelligent IT Operations Using Machine Learning," IEEE Access, vol. 10, pp. 112345–112360, 2022.
11. X. Xu et al., "AI-Based Fault Detection and Self-Healing in Cloud Systems," IEEE Trans. Cloud Computing, vol. 11, no. 3, pp. 1456–1468, 2023.
12. L. Wang et al., "Autonomous Cloud Management Using Reinforcement Learning," IEEE Cloud Computing, vol. 9, no. 2, pp. 34–43, 2022.
13. P. Chen et al., "Machine Learning for IT Operations (AIOps): A Review," ACM Comput. Surveys, vol. 53, no. 5, pp. 1–35, 2021.
14. Y. Liu et al., "Log-Based Anomaly Detection for Cloud Systems Using Deep Learning," IEEE Access, vol. 9, pp. 98765–98780, 2021.
15. B. Burns et al., "Kubernetes: Up and Running," O'Reilly Media, 2021.
16. Google Cloud, "Site Reliability Engineering: How Google Runs Production Systems," O'Reilly Media, 2021.