A Comparative Literature Review of Built-in AI-Powered Incident Response Agents Across AWS, Microsoft Azure and Google Cloud Platform

Kaushik Joshi, Hemant Patil, Harsh Patel, Pravin Wavare

Dr. D. Y. Patil College of MCA and Management Akurdi, Pune
Corresponding author email: joshikaushik419@gmail.com

Abstract — In today's cloud Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP) environments, traditional incident response becomes cumbersome, and vulnerable to human error. While Artificial Intelligence for IT Operations (AIOps) helped with machine learning in the area of anomaly detection, it remained advisory. The recent advances in Large Language Models (LLMs) and Agentic AI enable the construction of cloud architectures that can self-repair and function autonomously.With the recent advancements in Large Language Models (LLMs) and Agentic AI, it becomes possible to develop cloud architectures capable of self-healing and self-operationalizing. While there are many works that introduce a theoretical approach, or review a single vendor's built-in AI incident-response agent, no empirical study directly compares the built-in AI incident-response agents of the major hyperscalers in a similar workload and using the same fault injections. In AWS, Azure SRE Agent and AWS DevOps Agent, and in Azure and GCP, Azure SRE Agent and GCP Security AI, a standardized application (cloud_incident_tester) is deployed to allow for a fair comparison of the two agents. Detection Latency, Investigation Wall Clocks Time, Root Cause Analysis (RCA) accuracy, Tool execution error recovery, and Governance guardrails are what we measure. Azure SRE Agent reduces autonomous resolution time (3m 40s - 5m 00s), but needs to have a binding response plan, and will incur continuous billing costs of $0.40/hr. AWS DevOps Agent provides a deep investigative capability for a multi-region infrastructure with parallel subagents (9m 30s to 15m 23s), and will identify misconfigurations to the infrastructure without relying on a governance model – it just proposes them. The difference between GCP and other autonomous agents is that GCP does not provide any individual agent for users, but only enterprise organization security features with AI. The findings provide useful reference points for technology leaders and reveal configuration issues at a system level that could affect the reliability of a cloud.

Keywords — AI Agents, AIOps, Autonomous Systems, Cloud Computing, Incident Response , Native AI Agents

I. INTRODUCTION

The cloud management landscape is shifting from manual, rule-based operations to agentic AI orchestration. As hyperscale environments grow in complexity, comprising distributed microservices, ephemeral containers, and vast regional architectures, the efficacy of traditional incident management has plateaued. Static alerting thresholds and manual engineer interaction often result in slow Mean Time to Resolve (MTTR) metrics and significant cognitive load on on-call teams.

While early attempts at automation, collectively known as Artificial Intelligence for IT Operations (AIOps), introduced statistical machine learning for anomaly detection and log parsing, these platforms largely functioned as advisory services. They provided insights but required human operators to formulate diagnostic hypotheses and execute remediation steps. The advent of Large Language Models (LLMs) and advanced agentic AI frameworks represents a paradigm shift toward autonomous, self-healing architectures. By enabling semantic understanding of unstructured logs and dynamic API interaction, these modern agents promise to partition complicated responses into specialized tasks, such as monitoring, analysis, planning, and execution.

This study addresses the urgent need to evaluate the maturity of these proprietary offerings, specifically AWS DevOps Agent, Azure SRE Agent, and GCP Security AI, to understand their behavior, performance, and governance guardrails under controlled fault conditions in real-world hyperscaler environments. Furthermore, we examine the critical challenge of telemetry heterogeneity, where performance degrades as agents operate outside their primary training environments, providing a comprehensive assessment of how these hyperscaler agents perform in practice across diverse multi-cloud deployments. By synthesizing empirical fault-injection data with existing theoretical literature, this review identifies not only the operational strengths of these agents but also the systemic configuration pitfalls that remain largely unexamined in current commercial and academic discourse

II. RELATED WORK

1 Agentic AI's Emergence into a New Norm

There are three great streams of academic and industry literature when it comes to incident automation in the cloud: traditional AIOps, LLM-based agentic approaches, and a benchmarking exercise.Feedback loops were initially introduced in early autonomic computing frameworks [1] for the vision of self-managing systems. Initial deployments of AIOps were performed using deep learning architectures like DeepLog [3] and real-world event correlation models [4] to detect log anomalies. Key practical concerns identified were a large proportion of false alarms, lack of support for out-of-the-box log formats, and verbose logs. These limitations required human intervention to deal with incidents after the fact [2, 11].

2 Agents and Agent Components

Even when using a single agent, recent work has demonstrated that AI systems fail to perform end-to-end incident response correctly because they tend to hallucinate during diagnosis and hence produce inaccurate repairs [8, 12]. This limitation has been addressed by adding role-based frameworks and several multi-agent frameworks.IRCopilot introduced the dialogic support feature with an LLM to support on-call engineers, referred to as "ChatGPT2Call" [5]. Logically, the tasks in AIDR-Cloud [6] and AI Cloud Sentinel [7] are broken down into specific agents which perform different activities, such as Detector Agents that monitor telemetry, Diagnosis Agents that build topological dependency graphs, Planner Agents that build remediation steps, and Critic/Validator Agents that evaluate the safety of remediation steps.The empirical results demonstrate a much better actionable recommendation rate within multi-agent systems than in single-agent systems, with rates of up to 100% in special cases of multi-agent architectures and 1.7% [8, 9] in the latter. Furthermore, Infrastructure-as-Code and version-control repository integration enables agents to create pull requests and successfully address the request 70% of the time, according to the statistics [6, 10].

3 Security & Forensic Benchmarks

Benchmarks provide clearly and repeatably measurable evaluations of autonomous agents. SecRespond [13] and SIR-Bench [14] focus on post-compromise security incident response and triage, where agents were able to accurately identify security alerts with a high degree of accuracy (~97%) but were also prone to finding the same vector of security incidents repeatedly (alert parroting).There are two benchmarks to reason across cloud operations: OpenSec [15] and ORCA-bench [16], with the outputs of the latter showing poor accuracy as task complexity increases. In a medium-difficulty scenario of memory and network analysis, the accuracy of frontier models (Claude 3.5 Sonnet and GPT-4) reduces to 25.3% in real-world complex scenarios of Root Cause Analysis (RCA), as shown in SIABENCH [17] and the evaluation of Root Cause Analysis (RCA) [18].

III. METHODOLOGY

To ensure the scientific rigor and comparability of our results, we adopted a 6-step approach: Literature synthesis, Synthetic workload engineering, Multi-cloud deployment, Construction of the telemetry pipeline and Controlled empirical testing.

1. literature review and identification of gaps.

We grouped 20 key papers together into four categories: AIOps, self-healing systems, LLM orchestration, and security benchmarks [1–20]. This phase established empirical baseline and validated the lack of comparative literature that is native to the vendors.

2. Apply to Target Testbed Application & Fault Engine:

We created a microservice written in Python/Flask and put it into a multi-platform docker image (cloud_incident_tester) and the image (kkj1907/cloud-incident-tester:latest). The application provides three special endpoints to provide a simulation of various types of agent diagnostics:

- GET /api/error/log: returns deterministic 500 Internal Server Errors with python stack traces ranging from ZeroDivisionError to for parsing of logs.
- GET /api/error/resource: Starts multiple processes that use CPU for a certain time (default: 300s) to test resource exhaustion triage for all available vCPUs. ·
- GET /api/error/security: Simulates failed authentication bursts and attempts to access the system for which there is no permission to check ingest of audit logs.

3: Multi-Cloud Environment Deployment:

The test application was deployed to AWS, Azure and GCP, with containers being deployed using native infrastructure:

1. Amazon Web Services (AWS): Deployed on an EC2 instance t3.micro, using Docker and awslogs log driver to log to the CloudWatch Log Group /ecs/incident-tester. The monitoring was done by using CloudWatch metric filters and alarms (alert-http5xx, alert-cpu-high). The AWS DevOps Agent has been deployed in the us-east-1 region (Agent Space incident-tester-space) to discover across regions to us-east-2.

2. Microsoft Azure: Deployed to Azure Container Apps (eastus) with Azure Log Analytics workspace workspace-rgincidenttesteroXNW. For monitoring, rules for Azure

Monitor Log (alert-http5xx and alert-cpu-high) were added to an Action Group called ag-incident-alerts. A custom Response Plan (all-alerts-baseline) was used to bind Azure SRE Agent (sre- agent-research).

3. Google Cloud Platform (GCP): Deployed to Google Cloud Run (us-central1) with logs streaming to Cloud Logging. The metrics used for monitoring: custom log based metrics (http5xx_errors) and Cloud Monitoring Alerting Policies based on the following: CPU utilization, 5xx errors.

4: Comparative Evaluation Matrix: A comparative evaluation matrix was developed for the agents that covered five standardized dimensions: (1) Architecture & Autonomy, (2) Investigation Quality & Speed, (3) Data Integration & Telemetry Scope, (4) Performance & Error Recovery, and (5) Governance & Operational Safety.

5: Empirical Fault Injection Protocol: Experiments were done with control runs. All HTTP requests, error responses and time stamps were saved to a local a JSONL ground truth file (incident_responses.jsonl). Agents have been tested in both modes – prompt driven and autonomous – which is an unattended alert trigger.

6: Synthesis & Quality Control: All Agent outputs, Command logs, CloudTrail audit records, and App Insights traces were compared with ground truth files and the exact accuracy, wall clock time, and number of times that tools were executed were measured, as well as compliance with Governance.

IV. RESULTS AND DISCUSSION

The empirical analysis exposes that there are significant differences across the three analyzed cloud platforms on the four dimensions of autonomy, speed, investigation capability and governance.

1. **Azure SRE Agent:** performance metrics show that in the case of an HTTP 5xx error, there is no need for humans to intervene in the investigation, as the agent detects it autonomously within about 4.5 minutes, and investigates it in 3 minutes 40 seconds, with 100% accuracy across every incident which is measured. In the case of a CPU spike, the agent detects the error autonomously within about 8 minutes, and investigates it within 5 minutes, without any human intervention. With respect to governance, it has high level self healing abilities, handling things such as KQL syntax errors, permission issues, etc., but does not always trigger unapproved state changes, which requires human intervention. It also successfully identified the precise position of ZeroDivisionError exceptions, with the help of automated KQL queries, and showcased contextual governance.
2. **AWS DevOps Agent:** Achieves the same accuracy of RCA (100%), detects HTTP 5xx errors (23s vs 41s) and CPU spikes (5m vs 15m 23s), but also has a longer investigation time and was able to detect actual configuration bugs. It follows the "proposed only" approach when writing, which ensures that it always complies with the governance standard by generating a change proposal instead of applying the change. It applies the dual-surface architecture, mapped infrastructure in 16 regions, and predicted the exhaustion of credit due to CPU spikes proactively, while being constantly managed.
3. **Google Cloud Platform:** GCP Security AI is an enterprise organization security feature, and autonomous incident response is not supported for individual sandbox accounts, and detection is based on alert firing after fix. The log ingestion worked as expected, though there were some misconfigurations that were not addressed in the log that involved suppressing the warning and not having any alerts for intentional faults at the system level.

V. FUTURE WORK

- Multi-modal AI agents integration in hybrid-cloud scenarios.
- Consistency between cross cloud diagnostic benchmarks.
- Use of explainable AI (XAI) in autonomous remediation.

VI. CONCLUSION

In this empirical research, we discover that this increase is not theoretical; function ranges differently across various cloud service providers supplying self-governing cloud AI incident reaction solutions. The trade-offs of Azure SRE Agent are that it has a quick investigation time (less than 5 minutes), is focused enough to follow up fast alert investigations ($0.40/hr cost of ongoing investigation), and is very quick to investigate scenarios from the onset. AWS DevOps Agent provides the most of the details of the investigation, cross-region visibility, and categorization of 2nd order defects within 9-15 minutes of the proposed system's pure governance model.Systemic Configuration Pitfalls are operational risks due to the failure of AWS regex filter, GCP aggregation default, Azure response-plan prerequisite, etc. that are hard to monitor. It is believed that watch-less pipelines can blow unexpectedly.When implementing an engineering cloud AI agent: Ensure all the write gates are only for the creation of proposals; set alarm

persistence rules with stealthy alarm writing syntax until it is fully understood; cross-validate writing of the proposal alarm syntax with the native engineering tools; and, last but not least, do not take for granted that AI will always provide the right answers when an incident occurs

VII. REFERENCES

1. Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. IEEE Computer.
2. Zhao, L., et al. (2019). Real-world challenges in AIOps: A systematic study. ICSE.
3. Du, M., et al. (2017). DeepLog: Anomaly detection and diagnosis through deep learning. CCS.
4. Ghosh, R., et al. (2023). AIOps for cloud reliability. IEEE Trans. Cloud Comput.
5. Chen, Y., et al. (2025). IRCopilot: LLM-assisted incident response. arXiv:2505.20945.
6. Kumar, A., et al. (2026). AIDR-Cloud: AI-driven incident response. IEEE SoutheastCon.
7. Patel, S., et al. (2026). AI Cloud Sentinel: Autonomous cloud monitoring and self-healing. ICICV.
8. Liu, X., et al. (2025). Multi-agent LLM orchestration for deterministic incident response. arXiv:2511.15755.
9. Wang, M., et al. (2026). Autonomous multi-agent SRE architectures using CrewAI. IEEE Access
10. Zhang, H., et al. (2025). IFSHM: Integrated LLM and deep reinforcement learning for self-healing microservices. arXiv:2506.07411.
11. Smith, K., et al. (2025). Beyond AIOps: Systematic review of autonomous cloud operations. ACM Comput. Surv.
12. Tan, J., et al. (2026). LLMs and Agentic AI for cloud reliability: A systematic literature review. IEEE Trans. Serv. Comput.
13. Davis, E., et al. (2026). SecRespond: Evaluating LLM agents in security incident triage. arXiv:2607.26791.
14. Wilson, B., et al. (2026). SIR-Bench: Benchmarking security incident response agents. arXiv:2604.12040.
15. Martinez, C., et al. (2026). OpenSec: Open benchmarks for cloud security automation. arXiv:2601.21083.
16. Lee, D., et al. (2026). ORCA-bench: Benchmarking AI agents for cloud operational reasoning. arXiv:2607.28545.
17. Gupta, F., et al. (2026). SIABENCH: Comprehensive evaluation of AI agents in network and memory forensics. arXiv:2603.06422.
18. Taylor, R., et al. (2026). Self-healing cloud systems: A critical survey and evaluation framework. ICIICE.
19. Anderson, T., et al. (2026). LLM multi-agent systems survey: Architectures, challenges, and cloud applications. Springer Comput. Sci. Rev.
20. Amazon Web Services, (2026). AWS DevOps Agent User Guide and Technical Reference. AWS Documentation.

© Year The Author(s). Published by the International Journal of Engineering and Techniques (IJET). This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. License: https://creativecommons.org/licenses/by/4.0/