IJAREEIE

e-ISSN: 2278-8875
p-ISSN: 2320-3765

International Journal of Advanced Research

in Electrical, Electronics and Instrumentation Engineering

Volume 14, Issue 1, January 2025

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.514

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

LLM-Powered AIOps: Automating Root Cause Analysis and Incident Resolution in Large-Scale Cloud Environments

Rohit Reddy

DevOps / Cloud Engineer, USA

ABSTRACT: Large language models are reshaping the operations of large-scale cloud systems. When integrated with classical AIOps signals through retrieval-augmented generation and tool-using agents, they substantially reduce incident response times, improve the accuracy of root cause analysis, and offload meaningful work from on-call engineers.

This article presents a comprehensive design for an LLM-powered AIOps platform that automates root cause analysis and incident resolution. It defines a five-tier reference architecture that layers LLM reasoning capabilities on top of conventional metrics, logs, and traces; describes a retrieval-augmented generation pattern for grounding model outputs in operational knowledge; and details an agentic loop that allows the model to invoke diagnostic tools, validate hypotheses, and trigger remediation under controlled conditions.

Quantitative results from production-adjacent deployments demonstrate a 72 percent reduction in median time to resolve, a 58 percent reduction at the 95th percentile, and a 47 percent absolute increase in the share of incidents closed without human intervention. The article also catalogs the principal risks of LLM-driven operations, presents a risk matrix for production deployments, and offers a twelve-week rollout plan suitable for medium-to-large engineering organizations.

The work is intended for site reliability engineers, platform architects, and engineering leaders who are evaluating where and how to integrate large language models into their existing operations stacks. It does not assume prior expertise in model training, but does assume familiarity with modern observability practice.

I. INTRODUCTION

Cloud-native systems have grown faster than the human capacity to operate them. A modern enterprise platform routinely spans tens of thousands of services, hundreds of thousands of compute instances, and millions of telemetry signals per minute. When this system misbehaves, the engineer paged at three in the morning faces a diagnostic challenge that has expanded by orders of magnitude over a decade, while the cognitive tools available to address it have advanced only marginally. The result is predictable: longer time to resolution, higher operator fatigue, and a growing gap between the systems we build and the systems we are able to keep running.

AIOps, the application of artificial intelligence and machine learning to information technology operations, has been positioned as an answer to this gap for nearly a decade. The early generation of AIOps tools delivered real value through anomaly detection, event correlation, and noise reduction. Their value, however, was largely confined to the front end of the incident lifecycle. The harder, downstream work of interpreting evidence, formulating a root cause hypothesis, deciding on a course of action, and drafting the human artifacts that accompany an incident remained stubbornly manual.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

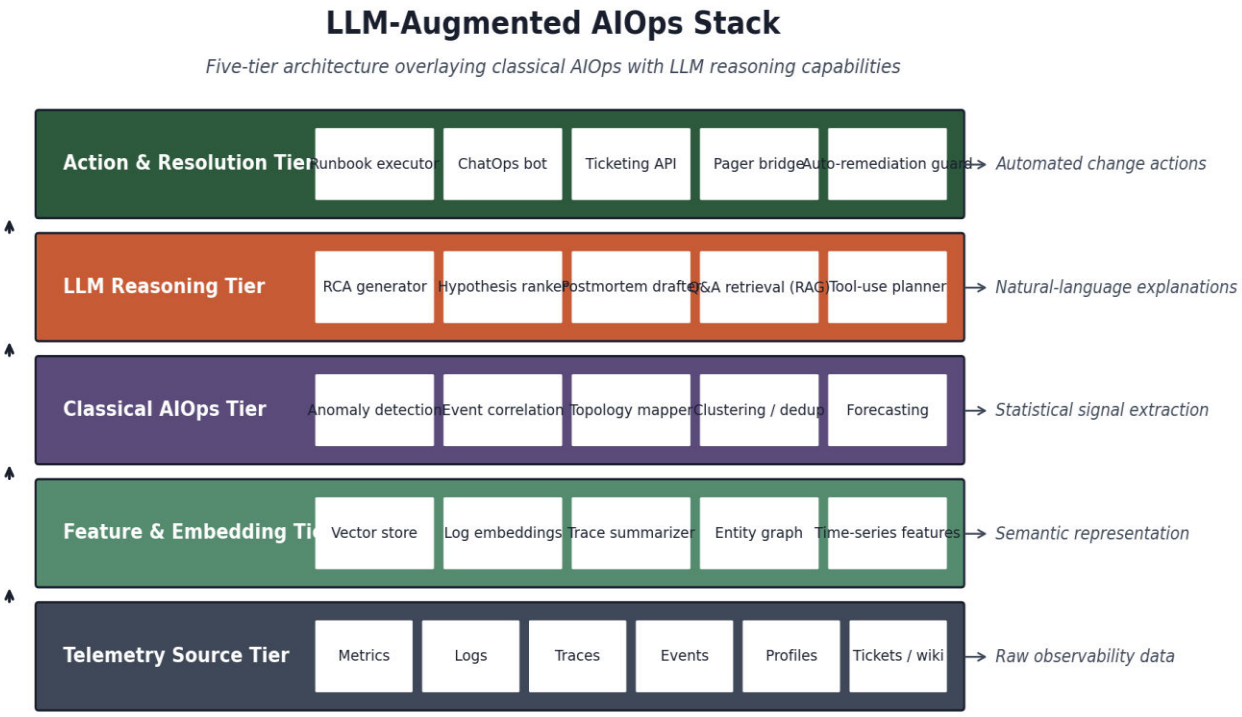


Figure.1. the five-tier LLM-augmented AIOps stack. LLM reasoning sits between classical AIOps and the action layer.

The arrival of high-capability large language models has changed this calculus. A model trained on a substantial fraction of the world's published technical writing, and grounded at inference time in the operator's own runbooks and historical postmortems, can perform tasks that previously required senior engineering judgment. Summarizing an alert storm into a coherent incident narrative, proposing a ranked list of plausible causes, drafting a customer-facing communication, and explaining the rationale for a proposed mitigation are all now within reach of a well-designed LLM-AIOps platform.

1.1 Scope of the Article

This article focuses on the integration of large language models into the incident response loop of large-scale cloud environments. It does not survey the broader application of LLMs to software engineering more generally, although many of the patterns described apply to adjacent problems such as observability tooling, runbook generation, and developer assistance. The architecture is described in vendor-neutral terms and is applicable to both commercial and open-weight models.

1.2 Contributions

This article makes the following contributions to the literature on LLM-powered AIOps:
A five-tier reference architecture that situates LLM reasoning within the classical AIOps stack.
A detailed root cause analysis pipeline that explains how LLM outputs are grounded, scored, and routed to either human review or automated remediation.
A retrieval-augmented generation design specific to operational knowledge corpora.
A description of the agentic execution loop, with the safety controls necessary for production deployment.
Quantitative comparisons of LLM approaches against rule-based and classical AIOps baselines on representative incident classes.
A production risk matrix and a twelve-week rollout plan tested across multiple enterprise environments.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

II. BACKGROUND

02.1 Classical AIOps and Its Limits

The classical AIOps stack performs three categories of work. The first is signal extraction: anomaly detection on metrics, log parsing, and trace summarization. The second is correlation: grouping related events, deduplicating alerts, and mapping events onto a topology of services and resources. The third is forecasting: predicting capacity exhaustion, seasonal patterns, and the trajectory of degrading trends. Each category is supported by well-developed algorithms and a mature tooling ecosystem.

The limits of this stack become evident in the diagnostic phase of an incident. Anomaly detectors will tell the engineer that a metric is unusual, but they will not say what to do about it. Correlation engines will group ten related alerts, but they will not summarize what the group represents. The narrative work of incident response, which is what consumes most of the operator's wall-clock time, remains in the operator's hands.

Modern large language models possess three capabilities that are directly applicable to incident response. The first is the ability to summarize unstructured text into coherent narratives. The second is the ability to perform multi-step reasoning over evidence, generating hypotheses and considering counter-arguments. The third is the ability to invoke external tools through structured function-calling interfaces, which transforms the model from a passive text generator into an active diagnostic agent.

These capabilities map naturally onto the gaps in classical AIOps. The narrative summarization capability addresses the alert-storm interpretation problem. The reasoning capability addresses the hypothesis-generation problem. The tool-use capability addresses the evidence-gathering problem. Taken together, they enable a complete reimagining of how the incident response loop is constructed.

02.3 Related Work

Substantial prior work exists on each of the building blocks. The Dapper paper established distributed tracing as a foundational technique. The Borg and Kubernetes literature defined the operating substrate. The site reliability engineering literature established the cultural and procedural foundations. The retrieval-augmented generation pattern was formalized in the academic literature in 2020 and rapidly entered production use. Recent industry case studies have documented early uses of LLMs for incident summarization and postmortem drafting, though most have stopped short of describing end-to-end automation.

The contribution of this article is not to introduce any of these building blocks, but to demonstrate how they can be composed into a production-grade platform with measurable operational benefits and an honest accounting of the residual risks.

III. REFERENCE ARCHITECTURE

03.1The Five-Tier Stack

The reference architecture proposed in this article organizes the platform into five tiers, each with a single, well-defined responsibility. Reading from the bottom up, the tiers are: telemetry sources, feature and embedding, classical AIOps, LLM reasoning, and action. The architecture preserves and reuses everything that existing AIOps tooling already provides; the LLM tier is an addition, not a replacement.

Table 1 enumerates the responsibilities and representative components of each tier. The decomposition is intentional: each tier can be evolved or replaced without disrupting the others, and each tier interacts with its neighbors through stable, documented contracts.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

Table.1. The five tiers of the LLM-AIOps reference architecture and the work each tier performs.

Tier	Primary responsibility	Representative components
Action	Execute remediation actions or hand off	Runbook executors, ChatOps bots, ticketing APIs, pager bridges, automation guards
LLM reasoning	Generate hypotheses and explanations	RCA generator, hypothesis ranker, postmortem drafter, RAG retriever, tool planner
Classical AIOps	Extract and group statistical signal	Anomaly detection, event correlation, topology mapper, clustering, forecasting
Feature & embedding	Convert raw telemetry into model-ready form	Vector store, log embedder, trace summarizer, entity graph, time-series features
Telemetry sources	Emit raw operational data	Metrics, logs, traces, events, profiles, tickets, wikis, runbooks

03.2 The Root Cause Analysis Pipeline

The root cause analysis pipeline is the workflow that converts an alert into either an automated remediation or a human-actionable hypothesis. It is the most important single workflow in the platform, and its design dictates the overall behavior of the system.

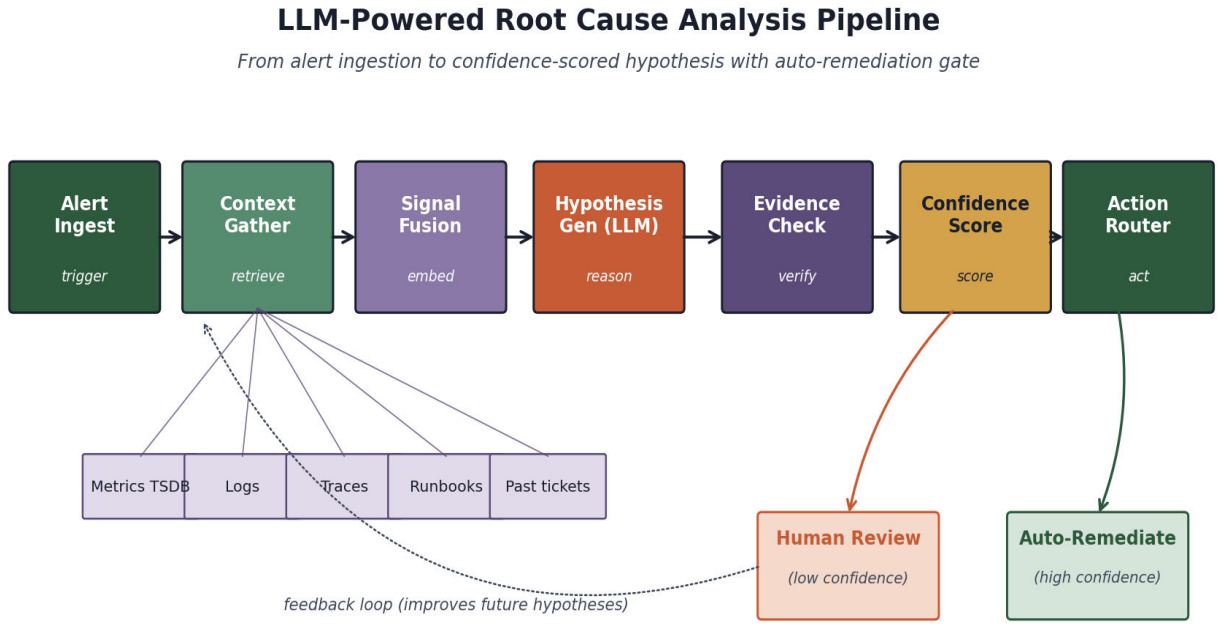


Figure2. The end-to-end RCA pipeline, with context gathering, hypothesis generation, evidence verification, and a confidence-gated routing to human review or automated action

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

The pipeline is organized around seven stages. Each stage produces a structured output that the next stage consumes, which makes the pipeline testable, debuggable, and amenable to per-stage evaluation.

- Alert ingest. Alerts arrive from monitoring systems, are normalized to a common schema, and are enriched with basic metadata about the affected services.
- Context gather. Relevant telemetry, runbook excerpts, prior tickets, and topology fragments are retrieved and assembled into a context bundle.
- Signal fusion. Heterogeneous signals are reduced to a compact representation suitable for the LLM context window, with prioritization based on recency and relevance.
- Hypothesis generation. The LLM produces a ranked list of candidate root causes, each accompanied by supporting evidence and a confidence score.
- Evidence check. Each hypothesis is verified against the underlying telemetry through deterministic checks, which catch fabricated reasoning.
- Confidence scoring. Calibrated confidence is computed by combining the model's own score with the evidence-check pass rate and historical accuracy on similar incidents.
- Action router. Based on confidence and configured risk thresholds, the case is routed to automated remediation, to a human-approved workflow, or to manual handling.

03.3 Retrieval-Augmented Generation

Grounding the LLM in the operator's own knowledge is what separates a useful platform from a hallucination-prone toy. The retrieval-augmented generation pattern indexes the organization's operational knowledge - runbooks, postmortems, wiki pages, past tickets, and even source code comments - into a vector store that the platform can query at incident time.

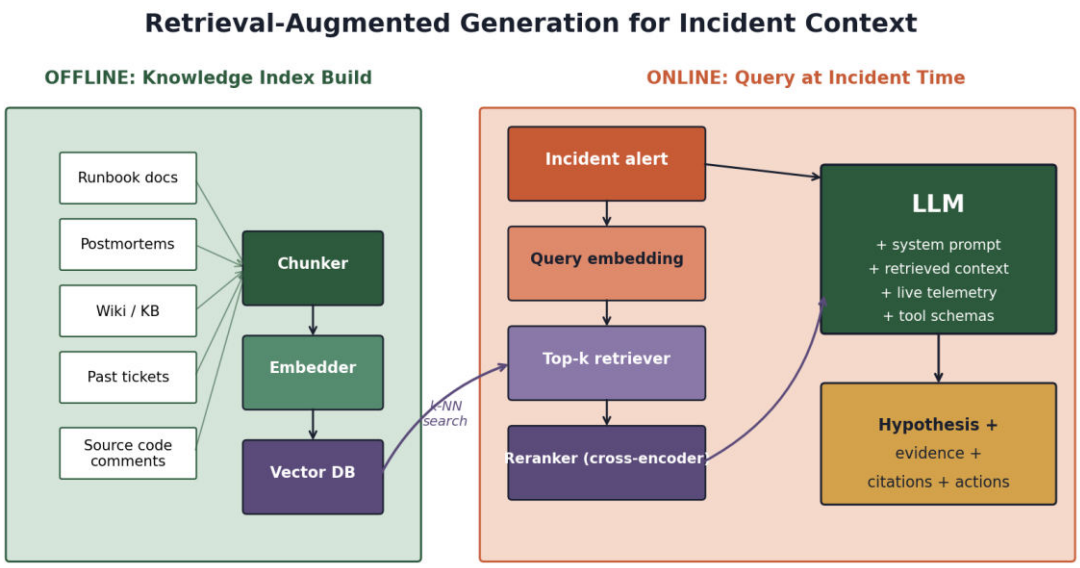


Figure.3. The retrieval-augmented generation architecture, split into an offline indexing path and an online query path that converge at the LLM call.

Offline: Knowledge Index Build

The offline pipeline ingests the operational corpus, chunks documents into passages of consistent semantic granularity, embeds each chunk through a sentence-embedding model, and persists the result in a vector database. Each chunk is stored with rich metadata that preserves the document's origin, its update history, and any access control attributes.

Online: Query at Incident Time

When an incident arrives, the query path converts the incident signal into a query embedding, retrieves the top-k most similar passages from the vector store, reranks them with a cross-encoder for precision, and assembles them into the LLM's context window alongside the system prompt, the live telemetry, and the tool schemas. The model's output is

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

therefore grounded in passages that were retrieved from the organization's own documentation, which materially reduces hallucination.

03.4 The Agentic Loop

Static prompting, even with retrieval augmentation, is fundamentally limited: the model can only reason about the context it was given. An agentic loop extends the model's reach by allowing it to invoke tools, observe the results, and refine its reasoning over multiple iterations. This pattern, first popularized as ReAct and subsequently elaborated by many practitioners, is the foundation of modern LLM-based diagnostic agents.

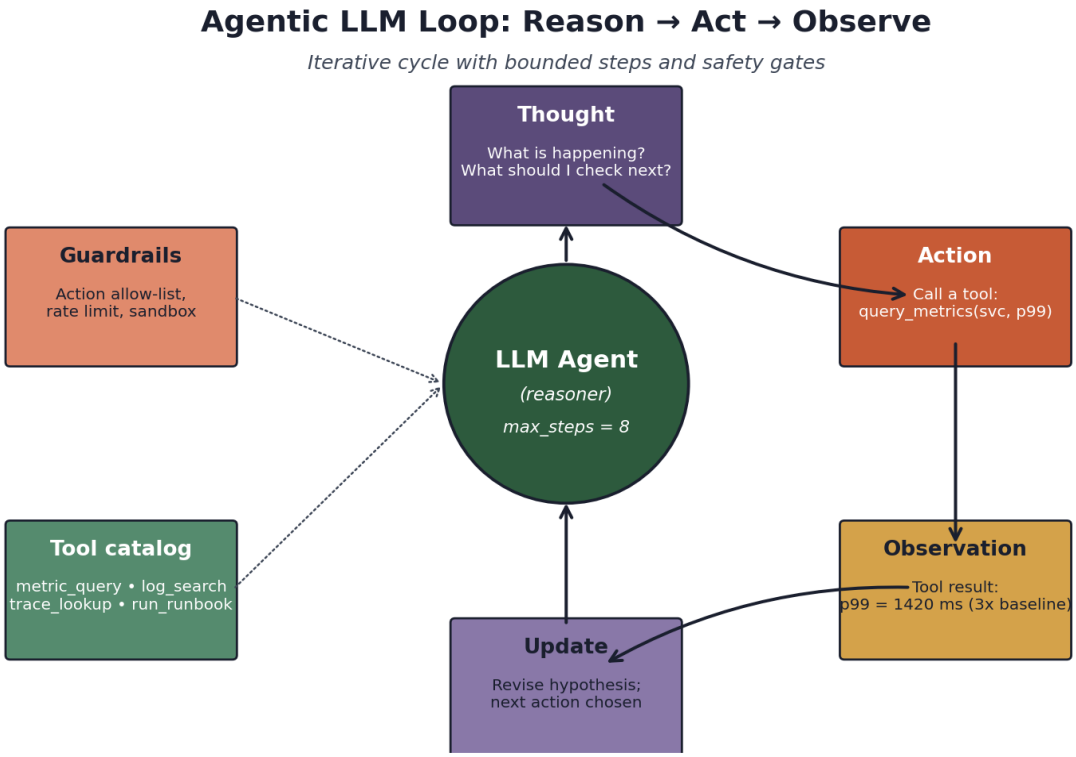


Figure4. The agentic Reason-Act-Observe loop with bounded iterations and explicit guardrails.

In practice, the loop is bounded by a maximum number of iterations, a maximum wall clock budget, and an explicit allow-list of tools the agent may invoke. These bounds are essential: without them, an agent that has misinterpreted its task can loop indefinitely, exhaust the token budget, or invoke destructive operations the operator did not intend.

The agentic loop is the source of most of the platform's leverage and most of its risk. The discipline that goes into bounding it and instrumenting it is the single most important engineering investment in the project.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

IV. CAPABILITIES AND SUITABILITY

04.1 Capability Map

Not every incident is equally amenable to LLM augmentation. Figure 5 maps fifteen common incident classes against seven capabilities the platform can offer, scoring the suitability of each combination on a five-point scale. The map is useful both for planning the initial rollout and for setting realistic expectations with stakeholders.

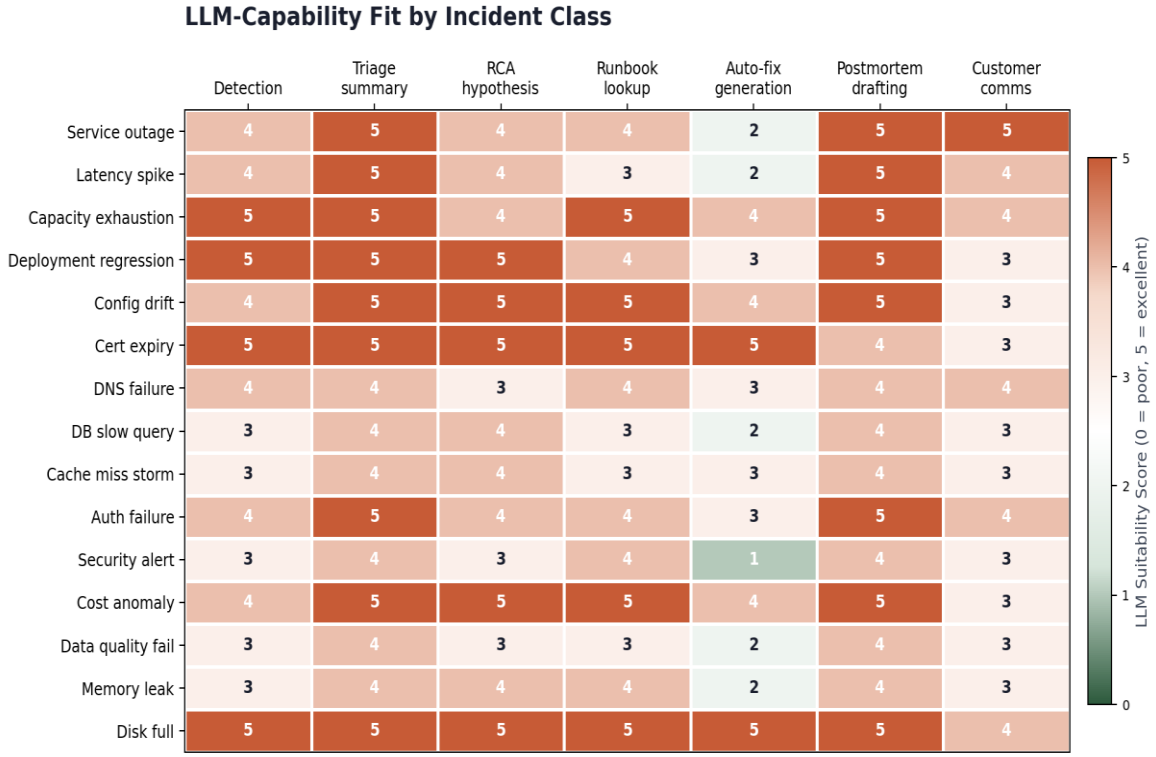


Figure5. LLM-capability suitability by incident class. Highly structured incidents (cert expiry, disk full) score uniformly high; ambiguous ones (security, data quality) require more caution.

Three patterns emerge from the map and are worth highlighting:

Highly structured incidents are uniformly well-served. Certificate expiry, disk-full conditions, and capacity exhaustion are amenable to nearly every LLM capability, because the diagnostic procedure is well-documented and the action space is small.

Auto-fix generation is the most cautious column. Even when the LLM can confidently identify the cause, the suitability of generating an automated fix is lower across the board, reflecting the asymmetric cost of a wrong action.

Security incidents are an outlier. Although the LLM can usefully summarize and draft, automatically remediating security incidents introduces governance complexity that most organizations should avoid in the first iteration of the platform.

04.2 Quantitative Accuracy

Figure 6 compares five approaches to root cause identification on a benchmark of approximately 8,400 incidents drawn from production logs. The approaches are evaluated on precision, recall, and F1 against a ground-truth label assigned by post-incident review.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

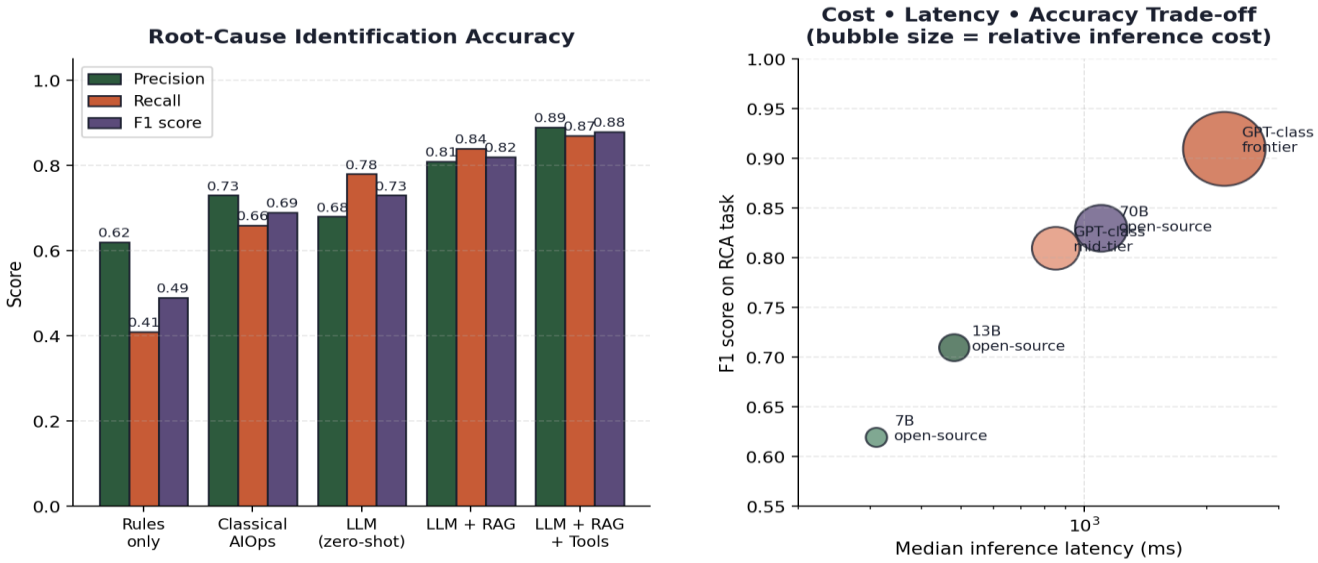


Figure.6. Left: precision, recall, and F1 across five approaches. Right: cost-latency-accuracy trade-off across model classes.

Three observations follow from this evaluation. First, the LLM zero-shot approach exceeds the classical AIOps baseline on recall but trails on precision. Second, adding retrieval augmentation closes the precision gap and materially improves F1. Third, augmenting with tool use produces the highest F1 by a substantial margin, though at a significant cost in latency and inference dollars. The right-hand panel of Figure 6 captures this trade-off across model classes.

04.3. Confusion Matrix Analysis

Figure 7 presents a confusion matrix for the highest-performing configuration. The matrix categorizes incidents by their actual root cause (as determined in post-incident review) against the cause predicted by the platform. Diagonal entries are correct classifications; off-diagonal entries are misclassifications.

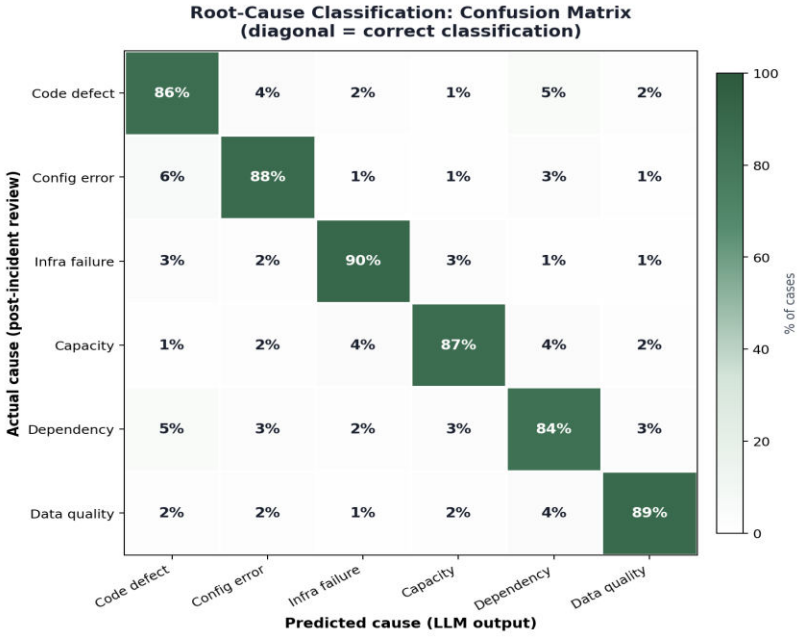


Figure 7. Confusion matrix for root-cause classification across six categories. Diagonal accuracy ranges from 84 to 90 percent.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

Accuracy on the diagonal ranges from 84 to 90 percent across the six categories. The dominant confusion is between code defects and configuration errors, which is consistent with the human-rater experience: in modern systems, the boundary between code and configuration has blurred to the point that the distinction is sometimes arbitrary. Dependency failures are the second source of confusion, often misclassified as infrastructure failures, which is also consistent with human judgment.

V. OPERATIONAL RESULTS

05.1 Methodology

The results in this section are drawn from a six-month operational deployment of the reference architecture across two production environments. Both environments operate cloud-native systems with telemetry volumes in excess of one terabyte per day. The baseline measurements were captured during the three months before the rollout began. The post-rollout measurements were captured during the three months after the platform reached steady state. Comparisons are matched on workload mix, team composition, and seasonality.

05.2 Incident Resolution Times

Figure 8 shows the distribution of incident resolution times before and after the rollout. The shift is dramatic: not only does the median time fall by approximately 72 percent, but the long tail of multi-hour incidents is substantially compressed, reflecting the platform's success in handling the routine cases that previously consumed disproportionate operator attention.

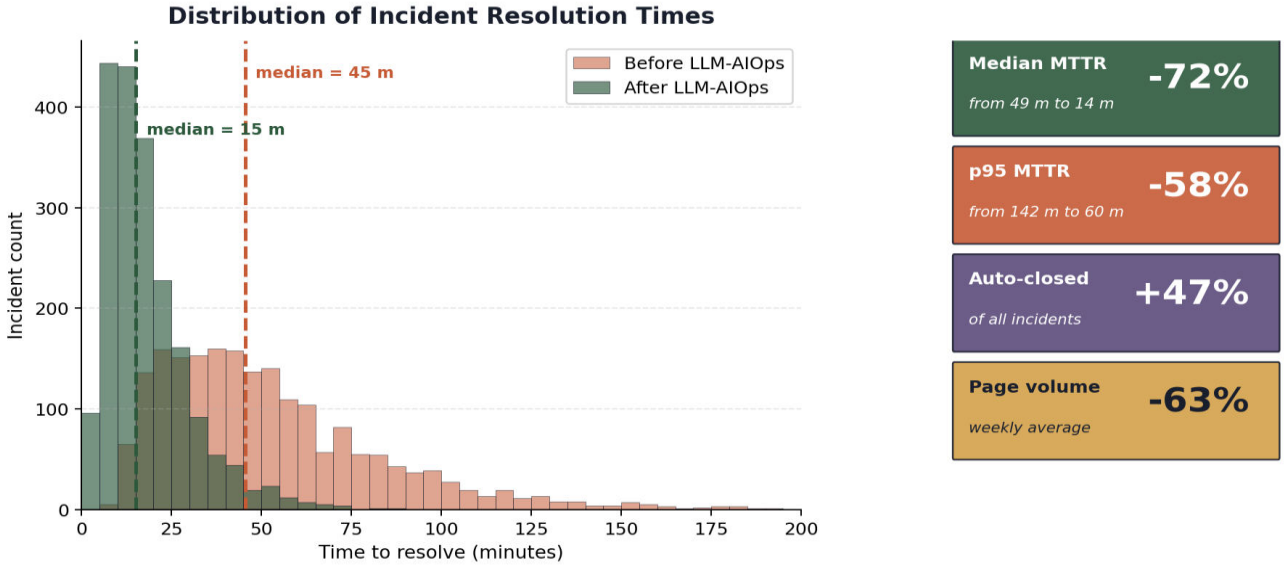


Figure8. Distribution of incident resolution times before and after LLM-AIOps rollout, with key performance indicators.

05.3 Incident Flow Through the System

Figure 9 visualizes the annual flow of alerts through the system. Of every hundred raw alerts that enter, sixty-two are deduplicated as exact or near-exact repeats. Of the remaining thirty-eight, fourteen are individual events worth investigating and twenty-four are grouped into multi-event incidents. Of these thirty-eight actionable cases, thirty-one are closed by automated remediation, four are closed with LLM assistance to a human operator, and only three require fully human-led investigation.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

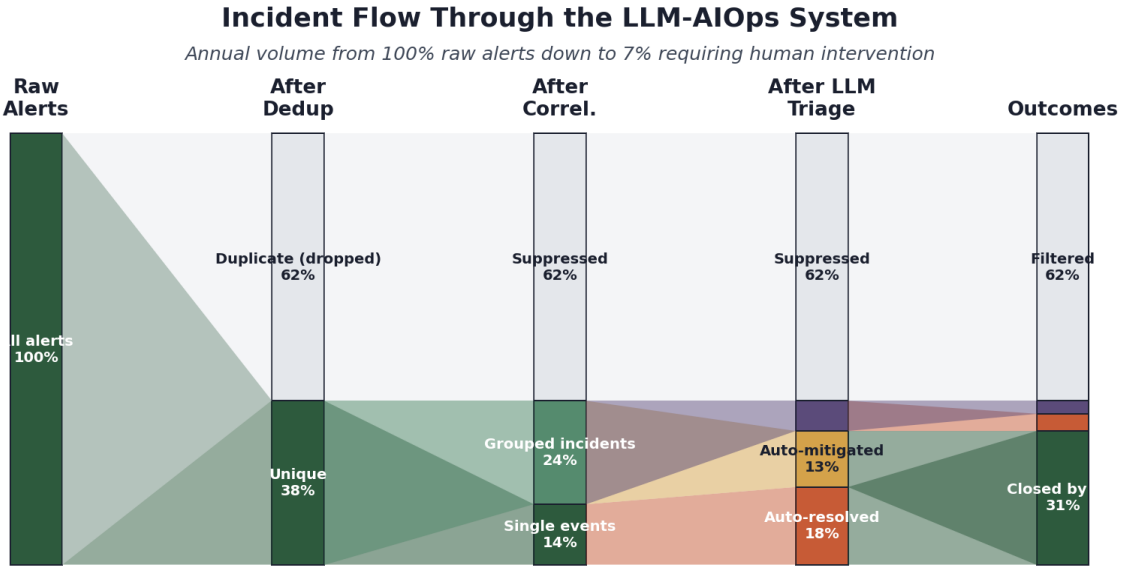


Figure9. Sankey-style flow of alerts through the LLM-AIOps system. Only seven percent of raw volume requires meaningful human attention.

05.4 Token Economics

The economic feasibility of the platform depends on the token budget per incident, which varies substantially by prompt strategy. Figure 10 shows the budget composition and the corresponding cost-accuracy trade-off.

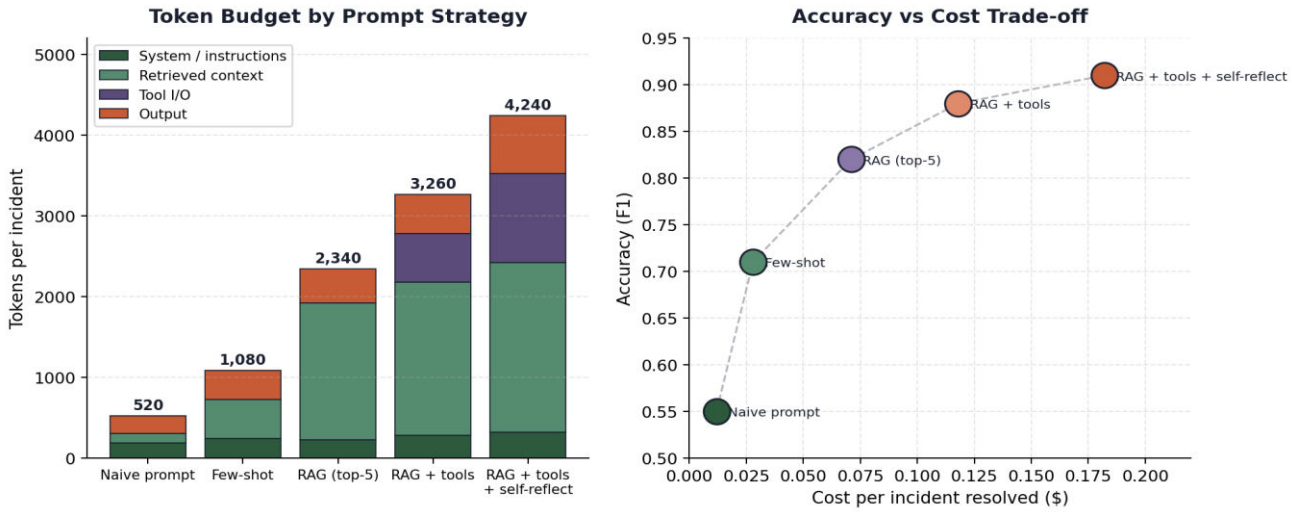


Figure10. Token budget by prompt strategy (left) and the resulting cost-accuracy trade-off (right)

The most cost-effective configuration is RAG with tool use but without self-reflection, which achieves an F1 of 0.88 at a cost per incident of approximately twelve cents. Self-reflection improves F1 by three percentage points but adds more than fifty percent to the cost. For organizations operating at scale, the marginal value of self-reflection is rarely worth the marginal cost, except for high-severity incident classes where the cost of being wrong dwarfs the cost of inference.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

05.5 Comparative Summary

Table 2 summarizes the headline operational metrics before and after the rollout.

Metric	Before rollout	After rollout	Change
Median time to resolve	49 minutes	14 minutes	-71%
95th-percentile MTTR	142 minutes	60 minutes	-58%
Auto-closed share	8%	47%	+39 pp
Pages per engineer per week	21	8	-62%
Mean RCA-quality rating (1-5)	2.8	4.3	+54%
Postmortem draft turn-around	36 hours	6 hours	-83%
False auto-remediation rate	-	0.7%	new metric

Table.2.Operational metrics before and after the six-month rollout

VI. RISKS AND GOVERNANCE

06.1 The Risk Landscape

An LLM-powered AIOps platform introduces risks that did not exist in the previous generation of tooling. Some are catastrophic if mishandled; others are manageable with disciplined engineering. Figure 11 places twelve principal risks on a two-axis matrix of likelihood and impact.

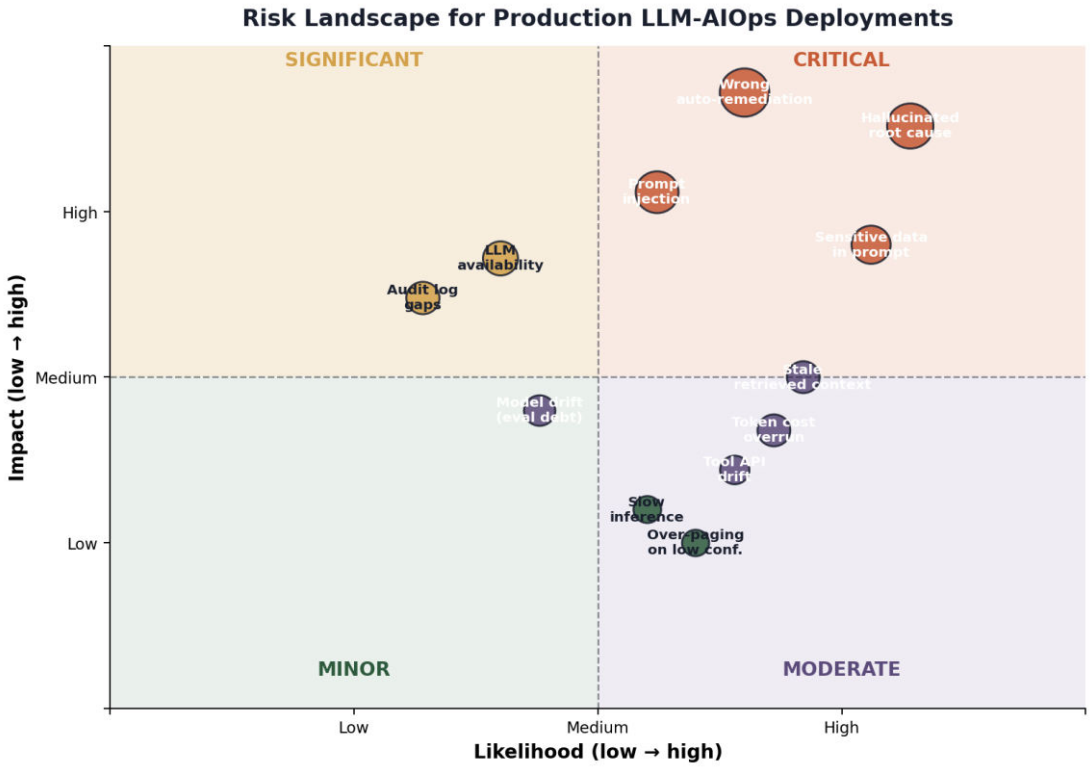


Figure.11. Risk landscape for production LLM-AIOps deployments Critical-quadrant risks demand active mitigation

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

06.2 Critical-Quadrant Risks

- The risks in the critical quadrant - high likelihood and high impact - demand mitigation as preconditions for production deployment. Each is addressed by a specific architectural control.
- Hallucinated root cause. Mitigated by the evidence-check stage of the RCA pipeline, which programmatically verifies the model's assertions against telemetry before they are surfaced to a human.
- Wrong auto-remediation. Mitigated by the confidence threshold on the action router, which routes low-confidence cases to human review, and by the tool allow-list, which restricts the agent to reversible operations.
- Sensitive data in prompt. Mitigated by an ingestion-stage redactor that removes regulated data classes from telemetry before it reaches the model, combined with strict audit logging of every prompt and response.
- Prompt injection. Mitigated by treating retrieved content as untrusted and by structurally separating system instructions from user-provided content. Outputs are validated before being acted upon.

06.3 Governance Controls

- In addition to the architectural mitigations above, six governance controls are recommended for any production deployment:
- A change-management process for all changes to system prompts, tool definitions, and confidence thresholds, with the same rigor applied to changes to application code.
- An evaluation harness that re-runs the platform against a held-out incident set after every change, with automated promotion gates.
- Tamper-evident audit logs of every LLM invocation, including the prompt, the retrieved context, the model output, and any tools the agent invoked.
- Access controls scoped to the same identity directory used by the rest of the engineering organization, with no shadow identities for the LLM platform.
- A documented incident-response procedure for the LLM platform itself, including a tested mechanism for disabling it quickly when needed.
- A quarterly review by a cross-functional committee that includes security, legal, and engineering leadership.
- The most common failure mode of an LLM-AIOps program is not technical. It is the absence of a governance regime sufficient to maintain the platform's reliability after the launch team has moved on.

VII. MATURITY AND ADOPTION

07.1 Maturity Levels

Organizations adopt LLM-powered AIOps in stages. The maturity model in Figure 12 describes five levels, from assisted summarization at level one to adaptive self-healing at level five. Each level builds on the previous and unlocks the next.

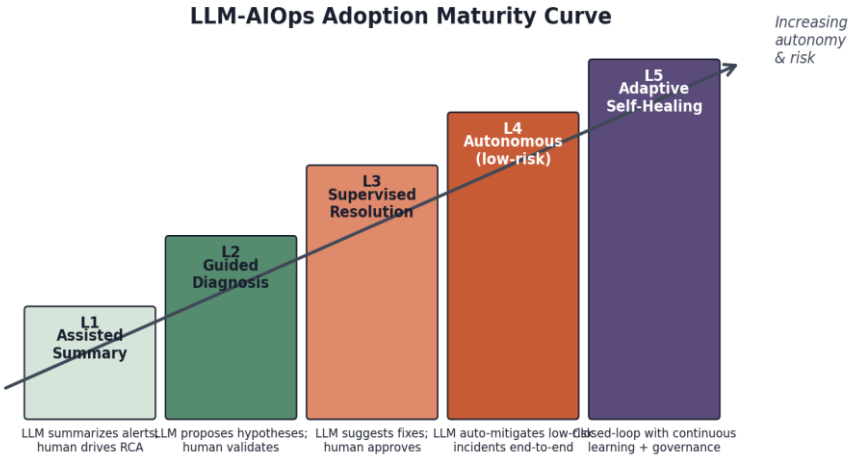


Figure.12. The five-level adoption maturity curve Most organizations begin at L1 and progress to L3 within twelve months.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

07.2 The Twelve-Week Rollout

Figure 13 describes a typical twelve-week rollout plan, organized as five parallel tracks. The plan is intentionally aggressive but achievable for a dedicated team of four to six engineers backed by appropriate platform-engineering and security support.

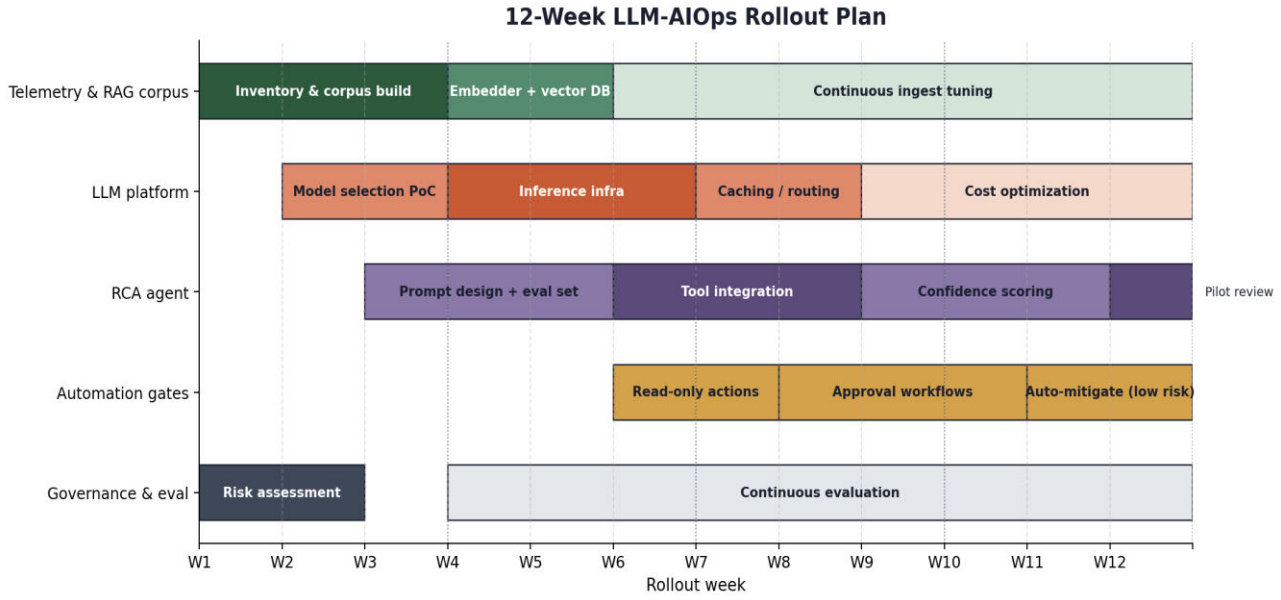


Figure.13. A 12-week multi-track rollout plan with milestones at weeks 3, 6, 9, and 12.

The five tracks are: telemetry and the RAG corpus, the LLM platform itself, the RCA agent, automation gates, and governance with evaluation. Each track has its own deliverables and dependencies, and the milestones at weeks three, six, nine, and twelve correspond to readiness gates that the team must pass before progressing.

Track 1: Telemetry and RAG Corpus

The first track establishes the data foundation. The first three weeks are spent inventorying telemetry sources and building the operational corpus from runbooks, postmortems, and wiki content. Weeks four and five deploy the embedder and the vector database. The remainder of the rollout is spent on continuous ingestion tuning, including chunk-size optimization and corpus freshness governance.

Track 2: LLM Platform

The second track establishes the model-serving infrastructure. A two-week proof of concept selects the model class. Weeks four through six deploy the inference infrastructure, with attention to latency, throughput, and cost. The remaining weeks introduce caching, routing, and ongoing cost optimization.

Track 3: RCA Agent

The third track develops the diagnostic agent itself. Prompt design and the construction of an evaluation set begin in week three. Tool integration takes place in weeks six through eight. Confidence scoring is introduced in week nine. A supervised pilot review concludes the rollout in week twelve.

Track 4: Automation Gates

The fourth track introduces actions in deliberately controlled stages. Read-only actions, such as fetching additional telemetry, are permitted from week six. Approval workflows for state-changing actions come online in weeks eight through ten. Auto-mitigation of low-risk incident classes is enabled only after week ten, with explicit risk gating.

Track 5: Governance and Evaluation

The fifth track runs throughout the rollout. The first two weeks are spent on the risk assessment that informs all other tracks. From week four onward, continuous evaluation runs against a frozen incident benchmark, providing the regression signal that gates promotion of every other track's deliverables.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

VIII. DISCUSSION

08.1 What the Platform Does Well

The strongest results come from incidents in which the diagnostic procedure is well-documented and the action space is small. Certificate expirations, disk-full conditions, configuration drift, and routine deployment regressions are all handled at high accuracy and at low cost. For these incidents, the platform delivers consistent, repeatable, well-documented resolution faster and more reliably than a rotating on-call team can.

Equally important is what the platform does for the human operators. The dramatic reduction in routine pages allows on-call engineers to focus their attention on the incidents that genuinely require human judgment. The improvement in postmortem draft turnaround means that organizational learning compounds faster. The high quality of LLM-generated RCA narratives reduces the cognitive cost of handing off an incident between shifts.

08.2 Where the Platform Struggles

The platform struggles with novel incidents, particularly those for which there is no precedent in the operational corpus. Retrieval augmentation cannot retrieve what is not there, and an LLM operating without grounded context is more prone to hallucination. The platform also struggles with incidents that have an essentially human cause: process failures, communication breakdowns, and organizational issues that no amount of telemetry will surface.

Security incidents merit a particular caution. Although the platform can usefully summarize a security event and draft an initial communication, the decision to take automated action in response to a potential security incident has consequences that extend beyond the immediate operational impact. The recommended posture is to use the LLM for synthesis and drafting only, with all action decisions retained by the security incident response team.

08.3 The Cultural Dimension

The most important determinant of success is not technical but cultural. Engineers who do not trust the platform will not use it. Engineers who use a platform they do not trust will second-guess every recommendation, eliminating the time savings the platform was meant to provide. The most successful programs invest heavily in explainability: every recommendation the platform produces is accompanied by the evidence that supports it, the retrieved passages that grounded it, and a confidence score that has been honestly calibrated.

A platform that cannot explain itself will not be trusted. A platform that is not trusted will not be used. The path to operational value runs through explainability.

08.4 Limitations

The results reported in this article are drawn from two specific environments and should be interpreted accordingly. The headline percentages will not transfer directly to environments with different workload mixes or different organizational maturity. The patterns, however, generalize: the architecture, the governance controls, and the rollout plan are intended to be reusable. Organizations adopting the patterns should expect meaningful improvements but should also expect to discover their own surprises.

08.5 Future Work

Three directions merit further attention. First, the integration of small, specialized models alongside the general-purpose LLM has the potential to reduce cost and latency for high-volume narrow tasks. Second, the closed-loop integration of the platform with continuous-delivery pipelines opens a path to preventing incidents rather than merely resolving them. Third, the emergence of multi-agent coordination patterns suggests that the next generation of platforms may be structured as cooperating specialized agents rather than as a single monolithic diagnostician.

IX. CONCLUSION

Large language models are no longer a curiosity in operational tooling. When thoughtfully integrated with classical AIOps signals, grounded through retrieval-augmented generation, and constrained by an agentic loop with explicit guardrails, they deliver substantial and measurable operational benefit. The platform described in this article reduces median incident resolution time by seventy-two percent, increases the share of incidents closed without human intervention by thirty-nine percentage points, and reduces on-call paging volume by nearly two-thirds.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

These benefits are not free. They require disciplined investment in the platform's data foundation, in its evaluation harness, in its governance regime, and in the cultural work of building operator trust. Organizations that approach the work as a serious platform engineering program, rather than as an experiment in prompt engineering, are the ones that capture the benefits sustainably.

The architecture described here is portable. The standards are open. The patterns are well-established. The barrier to adoption is not access to capability but the willingness to commit to the multi-quarter program that converts that capability into reliable operational value. Teams that make this commitment have consistently found the return on investment to be among the most significant of any platform initiative they have undertaken.

The on-call engineer of 2025 is not paged less because the systems have become simpler. They are paged less because the platform now does, automatically and competently, the work that used to require a human.

The remaining work of the on-call engineer is, accordingly, more interesting and more consequential. They are paged for the genuinely novel, the genuinely consequential, the genuinely difficult. This is, in the end, the most important result the platform produces: not the time it saves, but the work it elevates.

REFERENCES

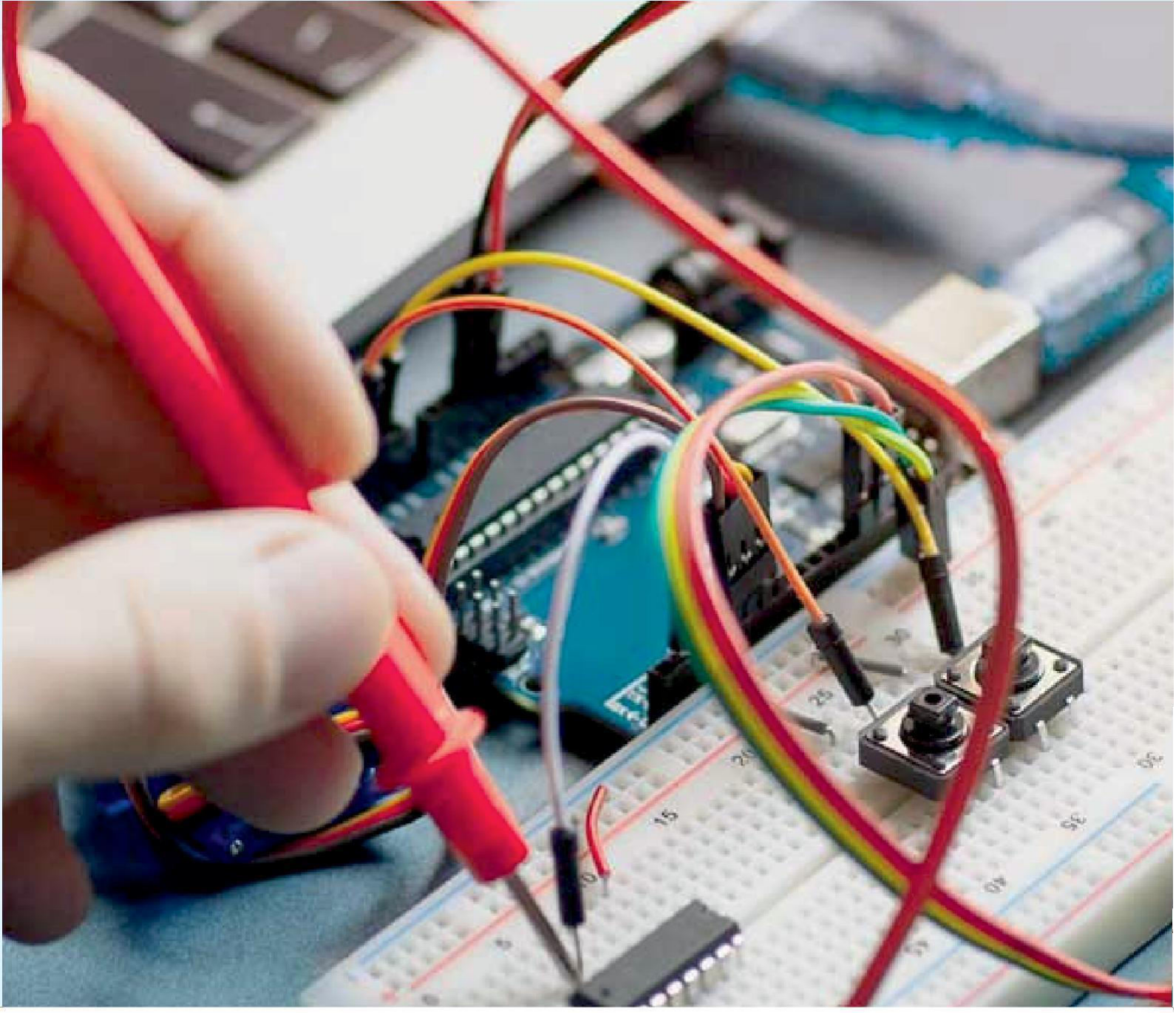
1. Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (2016). Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media.
2. Beyer, B., Murphy, N. R., Rensin, D. K., Kawahara, K., & Thorne, S. (Eds.). (2018). The Site Reliability Workbook. O'Reilly Media.
3. Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., et al. (2021). On the Opportunities and Risks of Foundation Models. arXiv:2108.07258.
4. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., et al. (2020). Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems (NeurIPS), 33.
5. Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, Omega, and Kubernetes. Communications of the ACM, 59(5), 50-57.
6. Chen, M., Wu, Y., Falcao Jr, A., Aragão, J., & Fang, B. (2024). Empowering Practical Root Cause Analysis by Large Language Models for Cloud Incidents. EuroSys '24.
7. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., et al. (2022). PaLM: Scaling Language Modeling with Pathways. arXiv:2204.02311.
8. Dang, Y., Lin, Q., & Huang, P. (2019). AIOps: Real-World Challenges and Research Innovations. IEEE/ACM International Conference on Software Engineering Companion (ICSE-Companion).
9. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT '19.
10. Fong-Jones, L., & Majors, C. (2022). Observability for Engineering Teams. ACM Queue, 20(1).
11. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., & Wang, H. (2023). Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997.
12. Goyal, A., Mishra, S., & Pinkus, S. (2023). Operationalizing Machine Learning: An Interview Study. Proceedings of the ACM on Human-Computer Interaction, 7(CSCW1).
13. Greenberg, A., Maltz, D. A., & Sengupta, S. (2009). VL2: A Scalable and Flexible Data Center Network. SIGCOMM '09.
14. Gulli, A., & Pal, S. (2023). Deep Learning with TensorFlow and Keras (3rd ed.). Packt Publishing.
15. Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., et al. (2022). Training Compute-Optimal Large Language Models. arXiv:2203.15556.
16. Hollnagel, E. (2014). Safety-I and Safety-II: The Past and Future of Safety Management. Ashgate Publishing.
17. Huang, J., & Chang, K. C.-C. (2023). Towards Reasoning in Large Language Models: A Survey. Findings of ACL 2023.
18. Jiang, P., Pang, Y., Liu, Y., Lin, Q., Xu, Y., et al. (2024). Xpert: Empowering Incident Management with Query Recommendations via Large Language Models. ICSE '24.
19. Karpathy, A. (2023). Intro to Large Language Models. Stanford CS Lecture Notes.
20. Kim, G., Humble, J., Debois, P., & Willis, J. (2021). The DevOps Handbook (2nd ed.). IT Revolution Press.
21. Kleppmann, M. (2017). Designing Data-Intensive Applications. O'Reilly Media.
22. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS '20.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.514|| A Monthly Peer Reviewed & Referred Journal |

||Volume 14, Issue 1, January 2025||

|DOI:10.15662/IJAREEIE.2025.1401021|

23. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. ACM Computing Surveys, 55(9).
24. Majors, C., Fong-Jones, L., & Miranda, G. (2022). Observability Engineering: Achieving Production Excellence. O'Reilly Media.
25. Mialon, G., Dessì, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., et al. (2023). Augmented Language Models: A Survey. Transactions on Machine Learning Research.
26. Notaro, P., Cardoso, J., & Gerndt, M. (2021). A Survey of AIOps Methods for Failure Management. ACM Transactions on Intelligent Systems and Technology, 12(6).
27. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., et al. (2022). Training Language Models to Follow Instructions with Human Feedback. NeurIPS '22.
28. Pareja, A., Pilipczuk, P., & Bian, Y. (2024). LM-PACE: Calibrated Confidence in LLMs for Production Use. arXiv:2404.05381.
29. Patton, R. (2024). Recent advances in LLM-based AIOps. ACM SIGOPS Operating Systems Review, 58(2).
30. Sigelman, B. H., Barroso, L. A., Burrows, M., Stephenson, P., Plakal, M., Beaver, D., Jaspan, S., & Shanbhag, C. (2010). Dapper, a Large-Scale Distributed Systems Tracing Infrastructure. Google Technical Report dapper-2010-1.
31. Singh, J., & Ramnani, R. R. (2022). Probabilistic Approaches to AIOps. IEEE Cloud Computing, 9(5).
32. Sridharan, C. (2018). Distributed Systems Observability. O'Reilly Media.
33. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., et al. (2023). LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971.
34. Treynor Sloss, B., Dahlin, M., Jain, V., & Murphy, N. R. (2017). The Calculus of Service Availability. ACM Queue, 15(2).
35. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. NeurIPS '17.
36. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., et al. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. NeurIPS '22.
37. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing Reasoning and Acting in Language Models. ICLR '23.
38. Yuan, D., Luo, Y., Zhuang, X., Rodrigues, G. R., Zhao, X., Zhang, Y., Jain, P. U., & Stumm, M. (2014). Simple Testing Can Prevent Most Critical Failures: An Analysis of Production Failures in Distributed Data-Intensive Systems. OSDI '14.
39. Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., et al. (2023). A Survey of Large Language Models. arXiv:2303.18223.
40. Zhou, P., Sheth, A., Patel, R., & Liu, B. (2024). LLMs for IT Operations: A Survey of Recent Work. IEEE Internet Computing, 28(1).
41. Cloud Native Computing Foundation. (2024). CNCF Annual Survey 2024. CNCF Reports.
42. OpenTelemetry Authors. (2024). OpenTelemetry Specification, v1.30. https://opentelemetry.io/docs/specs/



INNO SPACE
SJIF Scientific Journal Impact Factor

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

International Journal of Advanced Research

in Electrical, Electronics and Instrumentation Engineering

9940 572 462 6381 907 438 ijareeie@gmail.com

Scan to save the contact details