Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Operational Knowledge Graph Architecture for Explainable Artificial Intelligence in Cloud-Native DevOps Operations

Manvitha Potluri

24X7 Systems, USA

ARTICLEINFO

ABSTRACT

Modern artificial intelligence for IT operations (AIOps) platforms increasingly rely on large language models to assist with incident investigation, deployment reasoning, and operational decision support in cloud-native environments. Most existing systems, however, treat infrastructure telemetry, container orchestration state, continuous integration and deployment (CI/CD) metadata, monitoring signals, and historical incident records as disconnected data sources. This fragmentation limits the explainability of AI-generated recommendations and increases the risk of hallucinated or unsupported conclusions. The primary objective of this study is to design an Operational Knowledge Graph (OKG) architecture that unifies these heterogeneous data sources into a single semantic model capable of grounding large language model reasoning in verifiable infrastructure relationships. A design science research approach was followed, combining a structured review of knowledge graph-based root cause analysis and graph-grounded retrieval-augmented generation (GraphRAG) literature with layered architectural decomposition informed by operational experience with production Kubernetes and cloud infrastructure. The methodology produces a five-layer architecture spanning data ingestion, semantic modeling, temporal graph construction, GraphRAG-based retrieval and reasoning, and explainability. The architecture is demonstrated through two representative operational scenarios drawn from common cloud-native failure classes: a container image pull failure caused by a missing network path between private compute nodes and a container registry, and a load balancer provisioning failure blocked by an account-level policy boundary. In both scenarios, graph-grounded traversal produces provenance-backed causal narratives spanning infrastructure, orchestration, and governance layers, in contrast to the fragmented or unsupported explanations typical of log-only or LLM-only diagnostic approaches. The proposed architecture improves recommendation traceability, reduces ambiguity in AI-generated operational guidance, and provides a practical foundation for explainable, trustworthy AI-assisted operations in cloud-native DevOps platforms. Future work includes quantitative benchmarking against established AIOps datasets and integration with policy-as-code enforcement pipelines.

Keywords: Knowledge graph; AIOps, explainable artificial intelligence; Retrieval-augmented generation; Cloud-native DevOps; Root cause analysis; Kubernetes

1. INTRODUCTION

Cloud-native infrastructure has grown substantially in scale and complexity over the past decade, with organizations routinely operating hundreds of interdependent microservices across container orchestration platforms, managed cloud services, and continuous integration and deployment pipelines. This growth in operational complexity has outpaced the capacity of human operators to reason manually about system behavior, driving the widespread adoption of artificial intelligence for IT operations, commonly referred to as AIOps, as a discipline for automating anomaly detection, incident triage, and root cause analysis [1], [2].

The integration of large language models (LLMs) into AIOps platforms has expanded the scope of automation from narrow statistical anomaly detection toward natural-language incident investigation, deployment reasoning, and conversational operational assistance [3], [4]. Large language models are attractive for this purpose because they

can synthesize unstructured signals such as log messages, alert descriptions, and postmortem narratives into coherent explanations. This capability, however, introduces a well-documented risk: LLMs are prone to hallucination, generating plausible but factually unsupported statements when the information required to answer a query is absent from their context window or training data [5], [6].

In cloud-native operations, this hallucination risk is compounded by a structural problem rather than a purely algorithmic one. Infrastructure state, orchestration metadata, deployment history, and monitoring signals are typically stored in separate, purpose-built systems: cloud provider application programming interfaces (APIs) expose infrastructure state, the Kubernetes API exposes cluster and workload state, CI/CD platforms expose pipeline and release metadata, and observability platforms expose metrics, logs, and traces. Most AIOps systems query these sources independently and present the results to a language model as loosely structured text, discarding the relational structure that connects a failing pod to the node it runs on, the node to its network path, and the network path to a specific infrastructure configuration decision [7].

This paper addresses that structural gap by proposing an Operational Knowledge Graph (OKG) architecture that explicitly models cloud infrastructure, Kubernetes resources, CI/CD metadata, monitoring signals, and historical incidents as a unified, queryable graph. Rather than presenting a language model with disconnected text fragments, the proposed architecture retrieves and presents connected subgraphs, allowing the model to ground its reasoning in explicit, verifiable relationships and to cite the specific graph paths supporting a given conclusion. This design follows the broader graph-grounded retrieval-augmented generation (GraphRAG) paradigm, which has demonstrated promise for reducing hallucination and improving multi-hop reasoning in knowledge-intensive domains [8], [9], [10], and adapts it specifically to the operational semantics of cloud-native DevOps.

The primary contributions of this paper are threefold. First, it introduces a five-layer OKG architecture spanning data ingestion, semantic modeling, graph construction, GraphRAG-based reasoning, and explainability. Second, it defines a representative operational ontology covering infrastructure, orchestration, pipeline, and governance entities and the relationships connecting them. Third, it demonstrates the architecture through two representative operational scenarios illustrating how GraphRAG-based reasoning produces traceable, provenance-backed explanations for failure classes common in cloud-native environments. This paper follows a design science research approach, in which the problem is identified from a structured literature gap analysis, an artifact is designed to address that gap, the artifact is demonstrated through representative scenarios, and the outcome is evaluated against explicit qualitative design objectives, consistent with established design science research guidelines [11], [12].

The remainder of this paper is organized as follows. Section 2 reviews related work in knowledge graph-based root cause analysis and GraphRAG. Section 3 presents the proposed architecture. Section 4 discusses the representative operational scenarios and their implications and introduces an evaluation framework for future empirical work. Section 5 concludes the paper and outlines future work.

2. RELATED WORK

2.1. Root Cause Analysis in Cloud-Native and Microservice Systems

Root cause analysis in distributed systems has evolved through several methodological generations. Early approaches relied on statistical correlation and rule-based diagnosis, which struggled to scale with the dynamic topology of modern microservice deployments [13]. A subsequent generation introduced causal inference techniques to distinguish genuine causal relationships from spurious correlations in service dependency graphs, exemplified by CausalRCA, which applies causal discovery to fine-grained metric data to localize root causes in microservice applications [14]. Graph neural network approaches such as KGroot combine knowledge graph representations with graph convolutional networks to enhance root cause localization accuracy over purely statistical baselines [15]. Chain-of-Event constructs weighted event-causal graphs to provide interpretable, human-auditable explanations of failure propagation across services, addressing the opacity that limited earlier statistical methods [16]. A comprehensive survey of failure diagnosis methods in microservice systems documents this progression and identifies interpretability and generalizability as persistent open challenges across the field [2].

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

More recent work has begun incorporating large language models directly into the root cause analysis workflow. Roy et al. investigate the use of retrieval-augmented, tool-using LLM agents for root cause analysis of cloud incidents, finding that agentic approaches generalize to out-of-distribution production incidents better than static classifiers [3]. TAMO extends this direction with a tool-assisted agent reasoning over multi-modal observability data in cloud-native systems [17]. StateGraph combines Kubernetes cluster state graphs with LLM reasoning to simplify root cause analysis in containerized environments, which is closely related to the graph-grounded approach proposed here, though it focuses primarily on cluster-internal state rather than the broader infrastructure, pipeline, and governance context addressed in this work [7]. Chen et al. demonstrate that LLM-driven root cause summarization can substantially reduce operator time-to-diagnosis for complex cloud incidents when the underlying evidence is well organized [18].

2.2. Knowledge Graphs for Fault Diagnosis in Data Pipeline Systems

The generality of graph-based causal reasoning extends beyond cloud-native microservices; related work in adjacent operational domains applies similar principles to fault diagnosis. In industrial process monitoring, for example, a graph autoencoder combined with attention-based graph convolution has been used to detect faults in multivariate sensor data by incorporating the topological structure of a process plant alongside raw sensor readings, improving detection accuracy over sequence-only deep learning baselines [19]. A related line of work goes further by embedding causal intervention directly into the graph neural network architecture for fault diagnosis in complex industrial processes, where explicit causal graph structure is shown to be more interpretable than end-to-end learned models that offer no equivalent structural explanation [20]. Outside graph-based methods entirely, reinforcement learning has also been applied to self-healing data pipelines: Priyanka et al. combine deep Q-networks, proximal policy optimization, and meta-learning for real-time fault detection and autonomous recovery, a complementary direction in which remediation policy is learned from reward signals rather than derived from graph-derived state [21].

2.3. GraphRAG for Hallucination Mitigation in Knowledge-Intensive Reasoning

Outside the operations domain, GraphRAG has emerged as an active research area addressing hallucination in LLM outputs. Edge et al. introduce a GraphRAG framework that constructs community summaries over a knowledge graph to support query-focused summarization of large, unstructured private data corpora [22]. Subsequent work has refined GraphRAG along several dimensions, including path-pruning strategies that constrain retrieval to relevant relational paths [23] and lightweight graph retrieval architectures designed for practical deployment [24]. Surveys of hallucination in large language models identify insufficient grounding in verifiable external knowledge as a primary contributing factor and identify knowledge graph grounding specifically as a promising mitigation strategy because graph structure allows a system to expose the provenance of a claim as an explicit, traversable path rather than an opaque retrieval score [6], [25].

A parallel line of work addresses the construction of knowledge graphs themselves. Recent research explores LLM-assisted ontology construction pipelines that accelerate the extraction of domain-specific classes and relationships from unstructured enterprise data while preserving the logical structure required for downstream reasoning [26]. This is directly relevant to the operational domain, where infrastructure and governance semantics are heterogeneous across organizations and cloud providers, making a rigid, hand-authored ontology difficult to maintain at scale. The evolution and migration of CI/CD tooling further underscore the challenge of maintaining semantic models that stay current with rapidly changing operational environments [28], [29].

2.4. Gap Analysis

Reviewing this body of work reveals a consistent pattern: knowledge graph and causal graph approaches to root cause analysis have primarily modeled service dependency and metric relationships within a single observability domain, such as microservice call graphs or data pipeline lineage, while GraphRAG research has primarily targeted general-purpose knowledge-intensive question answering rather than the specific semantics of cloud infrastructure and DevOps governance. CI/CD metadata, cloud provider policy and quota constraints, and cross-layer relationships spanning networking, compute, and application layers are rarely represented within the same graph

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

as orchestration and monitoring state. This fragmentation limits the ability of AI-assisted operations tools to explain failures that originate at a layer boundary, such as a networking configuration decision that manifests as an application-layer symptom. Table 1 summarizes this positioning against three representative recent systems.

Table 1. Positioning of the Proposed OKG Architecture Against Representative Recent Approaches

Method	Logs / Metrics / Traces	Persistent Graph Scope	Governance-Aware	Explainability Mechanism
OpenRCA [27]	Yes (benchmark evidence bundles)	No persistent graph LLM localization over evidence bundles	No	LLM-generated rationale, not path-grounded
TAMO [17]	Yes (multi-modality: traces, metrics, logs)	Tool-assisted retrieval has no persistent cross-run graph	No	Tool-call trace, not graph provenance
StateGraph [7]	Partial (Kubernetes cluster state)	Kubernetes intra-cluster state graph only	No	Graph-grounded, scoped to cluster
Proposed OKG	**Yes**	**Full-stack graph: infrastructure, orchestration, pipeline, governance**	**Yes**	**Graph-grounded, cross-layer path provenance**

The OKG architecture proposed in this paper addresses this gap by explicitly unifying infrastructure, orchestration, pipeline, and governance entities within a single semantic model, extending the GraphRAG paradigm to the full operational surface of a cloud-native DevOps platform.

3. PROPOSED OPERATIONAL KNOWLEDGE GRAPH ARCHITECTURE

3.1. Design Principles

The proposed architecture follows four design principles derived from the gap analysis in Section 2. First, unification: infrastructure, orchestration, pipeline, monitoring, and incident data must be represented as a single connected graph rather than federated across separate query interfaces. Second, explainability by construction: every conclusion surfaced to an operator must be traceable to an explicit graph path, not merely a retrieval similarity score. Third, incremental freshness: the graph must reflect near-real-time infrastructure state rather than a static snapshot, since cloud-native environments change continuously. Fourth, governance-awareness: the graph must represent policy, quota, and access-control constraints as first-class entities, since a substantial fraction of operational failures in managed cloud environments originates from governance boundaries rather than application defects. The architecture is organized into five layers described in detail below.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

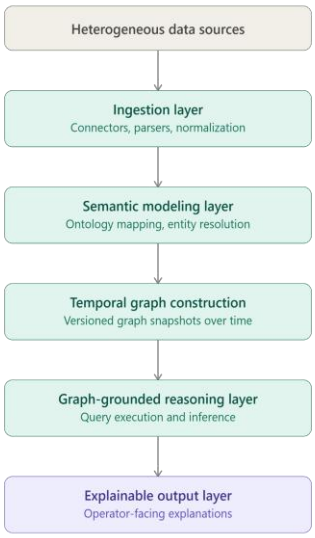

Figure 1. Five-layer OKG Architecture

3.2. Data Ingestion and Connector Layer

The ingestion layer maintains a set of source-specific connectors responsible for extracting entity and relationship data from operational systems of record. Cloud infrastructure connectors query provider APIs for compute, networking, identity, and quota state. Kubernetes connectors subscribe to the cluster API watch mechanism to capture workload, service, and endpoint state as it changes, rather than polling on a fixed interval. CI/CD connectors ingest pipeline run metadata, commit references, and deployment events from source control and delivery platforms [28], [29]. Monitoring connectors extract structured signals from metrics, log aggregation, and distributed tracing systems. Incident connectors ingest historical incident records and postmortem documents, which are parsed to extract structured entities such as affected services, contributing factors, and remediation actions. Each connector emits data in a common intermediate representation to decouple source-specific extraction logic from downstream graph construction.

3.3. Semantic Modeling Layer

The semantic modeling layer defines the operational ontology governing how ingested data is represented as graph entities and relationships. The ontology is organized into four entity families: infrastructure entities, including virtual private cloud (VPC), subnet, network endpoint, load balancer, and identity and access management (IAM) policy orchestration entities, including cluster, node, pod, deployment, service, and endpoint pipeline entities, including repository, pipeline run, build artifact, and deployment event and governance entities, including policy rule, service quota, and account-level configuration boundary. Table 2 summarizes representative relationship types connecting these entity families, enabling the graph to trace a causal path across layer boundaries.

Table 2. Representative Entity Relationship Types in the Operational Ontology

Relationship Type	From Entity	To Entity
routes_to	Service Endpoint	Pod
scheduled_on	Pod	Node
attached_to	Node	Subnet
connects_via	Subnet	Network Endpoint
contains	VPC Cluster	Subnet Node

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

fronts	Load Balancer	Service
deployed_by	Deployment	Pipeline Run
triggers	Repository	Pipeline Run
produces	Pipeline Run	Build Artifact
uses	Deployment	Build Artifact
constrained_by	Load Balancer	Service Quota
governed_by	Load Balancer	Policy Rule
instantiates	IAM Policy	Policy Rule
defines	Account Configuration Boundary	Service Quota
preceded	Incident	Deployment Event
affects	Incident	Service
correlates_with	Incident	Incident

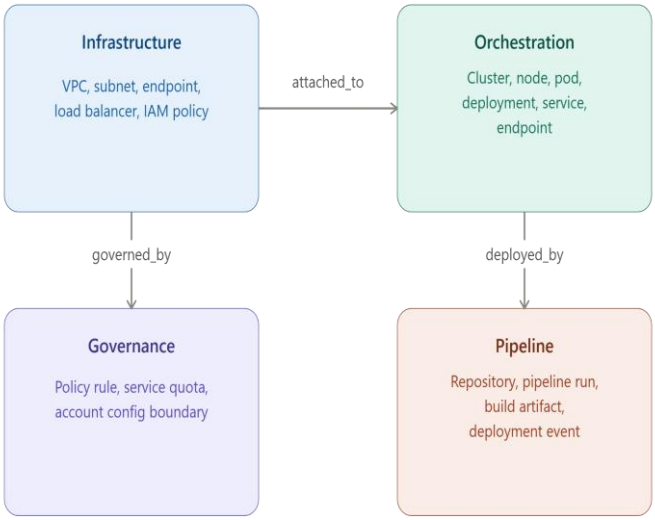

Figure 2. Operational Ontology Schema

3.4. Graph Construction and Temporal Versioning Layer

The construction layer transforms the intermediate representation emitted by connectors into graph mutations, applying the schema defined by the semantic modeling layer. Because cloud-native infrastructure changes continuously, this layer maintains a temporally versioned graph in which each edge and node property carries a validity interval, rather than a single mutable current-state graph. This design allows the reasoning layer to query both the current state of the system and its state at a specific past timestamp, which is essential for reconstructing the sequence of changes that preceded an incident. Change events are applied incrementally as they arrive from watch-based connectors, with periodic reconciliation passes against full-state snapshots to correct for missed or out-of-order events.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

3.5. GraphRAG-Based Retrieval and Reasoning Layer

When an operator or an upstream AIOps process submits a natural-language query, such as a request to diagnose a failing deployment, this layer first identifies the relevant seed entities, typically the affected pod, service, or deployment, and then performs constrained traversal outward from those seed entities along relationship types relevant to the query intent. This traversal produces a bounded subgraph rather than the entire graph, which is serialized into a structured representation and provided to the language model as grounding context, following the GraphRAG pattern established in prior work [22], [23]. Each retrieved path is assigned a confidence score based on the recency of the underlying evidence and the specificity of the relationship type, allowing the reasoning layer to prioritize direct causal paths over weaker correlational ones. This approach extends the entity-aware, path-pruned retrieval strategies developed in recent GraphRAG literature [24], [25] to the operational semantics of cloud-native infrastructure.

3.6. Explainability and Recommendation Layer

The final layer translates the LLM's grounded output into an operator-facing explanation. Rather than presenting a free-text narrative alone, this layer attaches the specific graph path or paths supporting each claim in the narrative, allowing an operator to inspect the underlying evidence directly. Where the retrieved subgraph includes governance entities such as a policy rule or service quota constraint, the recommendation layer surfaces the relevant policy context alongside the diagnosis, intended to reduce the frequency of recommendations that are technically plausible but organizationally infeasible. This layer also maintains a trust indicator based on the completeness of the supporting subgraph, distinguishing conclusions fully supported by direct graph paths from those requiring the language model to infer a connection not explicitly present in the graph. LLM-assisted schema induction techniques [26] are identified as a promising direction for extending the ontology to new providers without manual curation.

4. RESULTS AND DISCUSSION

4.1. Scenario One: Container Image Pull Failure Due to a Missing Network Path

A common failure class in managed Kubernetes environments occurs when nodes provisioned in a private subnet are unable to reach a container registry because no network path exists between the subnet and the registry endpoint, typically because a required private connectivity endpoint has not been provisioned. This failure manifests at the application layer as a container image pull error on the affected pods, which is the symptom an operator or an AIOps alerting system observes first. A log-only or metrics-only diagnostic approach typically surfaces this symptom directly, requiring the operator to manually trace backward from the failing pod through the node, the node's subnet, and the subnet's network configuration to identify the missing connectivity path. A language-model-only approach without graph grounding may correctly hypothesize this class of root cause based on the error message pattern, since it is a well-documented failure mode [7], but cannot confirm whether the specific missing connectivity path exists in the environment being diagnosed, which risks presenting a plausible but unverified conclusion.

Under the proposed architecture, the pod entity associated with the reported symptom is used as the traversal seed. The scheduled_on relationship connects the pod to its node, the attached_to relationship connects the node to its subnet, and the connects_via relationship connects the subnet to its configured network endpoints. Because the graph reflects the current infrastructure state, the absence of a connects_via edge to the required registry endpoint is directly observable as a missing relationship rather than an inferred hypothesis. The explainability layer surfaces this as a fully supported conclusion, citing the specific pod, node, and subnet entities involved, and distinguishes it from a lower-confidence hypothesis that would result if the connectivity path existed but a different symptom had been observed.

4.2. Scenario Two: Load Balancer Provisioning Blocked by an Account-Level Policy Boundary

A second common failure class occurs when a Kubernetes service configured to provision a network load balancer fails to complete provisioning because of an account-level constraint, such as a service quota limit or an IAM policy restriction, rather than a configuration error in the service specification itself. This failure class is particularly

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

difficult to diagnose from logs alone because the relevant error signal often originates from the cloud provider's control plane rather than from Kubernetes or application logs, and a language-model-only approach cannot distinguish a quota or policy boundary from a transient provisioning delay without direct access to account-level configuration state.

Under the proposed architecture, the front relationship connects the intended load balancer to its service, and the constrained_by and governed_by relationships connect the load balancer entity to the relevant service quota and policy rule entities ingested from the cloud provider's account configuration. When provisioning fails, traversal from the service entity surfaces the specific governance entity responsible for the block, and the correlates_with relationship allows the reasoning layer to identify prior incidents governed by the same policy boundary. This provides operators with organizational context, such as whether the same constraint has previously required an account-level support request to resolve, rather than a purely technical remediation step that would not address the underlying constraint.

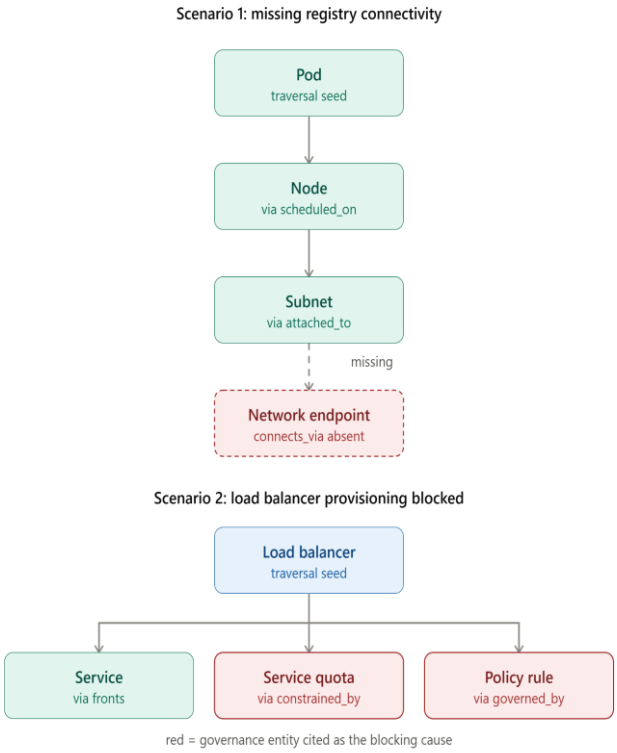

Figure 3. Traversal Paths for Scenarios 1 and 2

4.3. Discussion

Across both scenarios, the central benefit illustrated by the architecture is the conversion of an implicit, multi-hop diagnostic process, which an experienced operator would otherwise perform manually across several disconnected tools, into an explicit, traversable graph path that a language model can cite directly. This has two practical implications for explainability. First, because each claim in the generated explanation is anchored to a specific graph path, an operator can verify the claim independently rather than relying on the language model's assertion alone, directly addressing the hallucination risk identified in Section 1. Second, because the graph explicitly represents governance entities alongside infrastructure and orchestration entities, the architecture surfaces organizationally relevant context that a purely technical diagnostic approach would omit, a limitation identified in the gap analysis in Section 2 with respect to prior systems, including OpenRCA [27] and TAMO [17].

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

These scenarios also illustrate the architecture's limitations. The quality of the explanation is bounded by the completeness and freshness of the underlying graph, and a connector outage or a delay in incremental graph updates would degrade the confidence of the retrieval layer's output in a way that is detectable, given the confidence scoring described in Section 3.5, but not eliminated. Additionally, the ontology defined in Section 3.3, while representative, is not exhaustive, and organizations adopting this architecture would need to extend the entity and relationship schema to reflect their specific cloud provider services and internal governance structures. Future empirical work is required to quantify the traceability and hallucination-reduction benefits illustrated qualitatively here.

4.4. Evaluation Framework for Future Empirical Validation

Because this paper presents an architectural contribution rather than a trained model, no experimental results are claimed. To make the qualitative benefits illustrated in Sections 4.1 through 4.3 empirically testable in future work, Table 3 defines six evaluation dimensions directly derived from the design principles stated in Section 3.1, together with the intended outcome each dimension captures and a candidate metric a future study could compute against a labeled incident dataset and a non-graph-grounded baseline such as OpenRCA [27] or TAMO [17].

Table 3. Proposed Evaluation Dimensions for Future Empirical Validation

Dimension	Design Principle	Intended Outcome	Candidate Metric
Traceability	Explainability by construction	Provenance-backed explanations for every generated claim	Proportion of generated claims with a cited graph path
Hallucination reduction	Explainability by construction	Reduced unsupported or fabricated reasoning versus non-graph-grounded baselines	Rate of unsupported claims versus OpenRCA / TAMO baselines
Cross-layer reasoning	Unification	Correct diagnosis of failures originating at an infrastructure-orchestration-governance boundary	Proportion of correctly diagnosed incidents whose root cause spans more than one entity family
Graph freshness	Incremental freshness	Diagnosis reflects the current infrastructure state rather than a stale snapshot	Staleness interval between an infrastructure change and its graph update
Governance awareness	Governance-awareness	Correct identification of policy- or quota-blocked failures	Recall on a governance-boundary incident subset
Operator verifiability	Explainability by construction	An operator can independently verify a claim without relying on the model's assertion alone.	Operator time-to-verification citation sufficiency rate

5. CONCLUSION

This paper proposed an Operational Knowledge Graph architecture that unifies cloud infrastructure, Kubernetes orchestration state, CI/CD metadata, monitoring signals, and historical incident records into a single semantic model to ground large language model reasoning in AI-assisted cloud-native operations. The architecture was organized into five layers spanning data ingestion, semantic modeling, temporally versioned graph construction, GraphRAG-based retrieval and reasoning, and explainability. Two representative operational scenarios, a container registry connectivity failure and a governance-blocked load balancer provisioning failure, illustrated how the architecture produces provenance-backed, cross-layer causal explanations that are difficult to obtain from log-only or language-model-only diagnostic approaches.

This work has several limitations motivating future research. The architecture has not yet been evaluated quantitatively against established AIOps benchmarks or in a live production deployment. Future work should implement the graph construction and reasoning layers on a production-grade graph database backend such as Neo4j or Amazon Neptune, standardize telemetry ingestion around the OpenTelemetry specification to reduce connector-specific engineering effort, and measure diagnostic accuracy, latency, and hallucination rate against existing root cause analysis benchmarks such as the evidence bundles used in the OpenRCA evaluation [27] or the microservice incident datasets released for the CCF AIOps Root Cause Analysis Challenge, relative to non-graph-grounded baselines including TAMO [17]. Future work should also explore automated ontology extension using LLM-assisted schema induction [26] to reduce the manual effort required to adapt the architecture to new cloud providers. Finally, integrating the explainability and recommendation layer directly with policy-as-code enforcement pipelines represents a promising direction for closing the loop between explainable diagnosis and automated, policy-compliant remediation in cloud-native DevOps platforms.

ACKNOWLEDGMENTS

Not applicable.

FUNDING INFORMATION

The authors state no funding is involved.

AUTHOR CONTRIBUTIONS STATEMENT

Manvitha Potluri: Conceptualization, Methodology, Formal Analysis, Investigation, Writing – Original Draft Preparation, Writing – Review and Editing.

CONFLICT OF INTEREST STATEMENT

The authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] P. Kumar and H. S. Kumar, "AIOps: Analysing Cloud Failure Detection Approaches for Enhanced Operational Efficiency," in Proc. 2023 Int. Conf. on Applied Intelligence and Sustainable Computing (ICAISC), 2023, pp. 1-6, doi: 10.1109/ICAISC58445.2023.10199929. [Online]. Available: https://doi.org/10.1109/ICAISC58445.2023.10199929

[2] S. Zhang, S. Xia, W. Fan, B. Shi, X. Xiong, Z. Zhong, M. Ma, Y. Sun, and D. Pei, "Failure Diagnosis in Microservice Systems: A Comprehensive Survey and Analysis," arXiv:2407.01710, 2025. [Online]. Available: https://arxiv.org/abs/2407.01710

[3] D. Roy, X. Zhang, R. Bhave, C. Bansal, P. Las-Casas, R. Fonseca, and S. Rajmohan, "Exploring LLM-Based Agents for Root Cause Analysis," in Companion Proc. 32nd ACM Int. Conf. on the Foundations of Software Engineering (FSE), 2024, pp. 208-219. [Online]. Available: https://arxiv.org/abs/2403.04123

[4] R. D. Zota, C. Barbulescu, and R. Constantinescu, "A Practical Approach to Defining a Framework for Developing an Agentic AIOps System," Electronics, vol. 14, no. 9, 2025, doi: 10.3390/electronics14091775. [Online]. Available: https://doi.org/10.3390/electronics14091775

[5] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of Hallucination in Natural Language Generation," ACM Computing Surveys, vol. 55, no. 12, pp. 1–38, 2023, doi: 10.1145/3571730. [Online]. Available: https://doi.org/10.1145/3571730

[6] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu, "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions," ACM Transactions on Information Systems, 2025, doi: 10.1145/3703155. [Online]. Available: https://doi.org/10.1145/3703155

[7] Y. Xiang, C. P. Chen, L. Zeng, W. Yin, X. Liu, H. Li, and W. Xu, "Simplifying Root Cause Analysis in Kubernetes with StateGraph and LLM," arXiv:2506.02490, 2025. [Online]. Available: https://arxiv.org/abs/2506.02490

[8] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Advances in Neural Information Processing Systems 33, 2020, pp. 9459–9474. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[9] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li, D. Yin, T. Chua, and Q. Li, "A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models," in Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining, 2024, pp. 6491–6501. [Online]. Available: https://doi.org/10.1145/3637528.3671470

[10] Y. Hu, Z. Lei, Z. Zhang, B. Pan, C. Ling, and L. Zhao, "GRAG: Graph Retrieval-Augmented Generation," arXiv:2405.16506, 2024. [Online]. Available: https://arxiv.org/abs/2405.16506

A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," MIS Quarterly, vol. 28, no. 1, pp. 75–105, 2004, doi: 10.2307/25148625. [Online]. Available: https://doi.org/10.2307/25148625

[11] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," Journal of Management Information Systems, vol. 24, no. 3, pp. 45-77, 2007, doi: 10.2753/MIS0742-1222240302. [Online]. Available: https://doi.org/10.2753/MIS0742-1222240302

[12] J. Lin, P. Chen, and Z. Zheng, "Microscope: Pinpoint Performance Issues with Causal Graphs in Micro-Service Environments," in Proc. Int. Conf. on Service-Oriented Computing (ICSOC), 2018, pp. 3–20, doi: 10.1007/978-3-030-03596-9_1. [Online]. Available: https://doi.org/10.1007/978-3-030-03596-9_1

[13] R. Xin, P. Chen, and Z. Zhao, "CausalRCA: Causal Inference Based Precise Fine-Grained Root Cause Localization for Microservice Applications," Journal of Systems and Software, vol. 203, 2023, doi: 10.1016/j.jss.2023.111724. [Online]. Available: https://doi.org/10.1016/j.jss.2023.111724

[14] T. Wang, G. Qi, and T. Wu, "KGroot: Enhancing Root Cause Analysis through Knowledge Graphs and Graph Convolutional Neural Networks," arXiv:2402.13264, 2024. [Online]. Available: https://arxiv.org/abs/2402.13264

[15] Z. Yao, C. Pei, W. Chen, H. Wang, L. Su, H. Jiang, Z. Xie, X. Nie, and D. Pei, "Chain-of-Event: Interpretable Root Cause Analysis for Microservices through Automatically Learning Weighted Event Causal Graph," in Companion Proc. 32nd ACM Int. Conf. Foundations of Software Engineering (FSE), 2024, pp. 50–61. [Online]. Available: https://dl.acm.org/doi/10.1145/3663529.3663827

[16] X. Zhang, Q. Wang, M. Li, Y. Yuan, M. Xiao, F. Zhuang, and D. Yu, "TAMO: Fine-Grained Root Cause Analysis via Tool-Assisted LLM Agent with Multi-Modality Observation Data in Cloud-Native Systems," arXiv:2504.20462, 2025. [Online]. Available: https://arxiv.org/abs/2504.20462

[17] Y. Chen, H. Xie, M. Ma, Y. Kang, X. Gao, L. Shi, Y. Cao, X. Gao, H. Fan, M. Wen, J. Zeng, S. Ghosh, X. Zhang, C Zhang, Q. Lin, S. Rajmohan, D. Zhang and T. Xu, "Automatic Root Cause Analysis via Large Language Models for Cloud Incidents," arxiv: 2305.15778, 2023, doi: https://doi.org/10.1145/3627703.3629553 [Online]. Available: https://arxiv.org/pdf/2305.15778

[18] P. Brahmbhatt, R. Patel, A. Maheshwari, and R. D. Gudi, "Improved Fault Detection and Diagnosis Using Graph Auto Encoder and Attention-based Graph Convolution Networks," Digital Chemical Engineering, vol. 11, 100158, 2024, doi: 10.1016/j.dche.2024.100158. [Online]. Available: https://doi.org/10.1016/j.dche.2024.100158

[19] R. Liu, Q. Zhang, D. Lin, W. Zhang, and S. X. Ding, "Causal Intervention Graph Neural Network for Fault Diagnosis of Complex Industrial Processes," Reliability Engineering & System Safety, vol. 251, 110328, 2024, doi: 10.1016/j.ress.2024.110328. [Online]. Available: https://doi.org/10.1016/j.ress.2024.110328

[20] K. Priyanka, S. Priyadharshini, N A. Vignesh, A. Hameed, R. Reddy K, and Praveenkumar C, "Self-Healing Data Pipelines: Reinforcement Learning for Real-Time Fault Detection and Autonomous Recovery," International Conference on Metaverse and Current Trends in Computing (ICMCTC), 2025, doi:

10.1109/ICMCTC62214.2025.11196544. [Online]. Available: https://ieeexplore.ieee.org/document/11196544

[21] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, and J. Larson, "From Local to Global: A Graph RAG Approach to Query-Focused Summarization," arXiv:2404.16130, 2024. [Online]. Available: https://arxiv.org/abs/2404.16130

[22] B. Chen, Z. Guo, Z. Yang, Y. Chen, J. Chen, Z. Liu, C. Shi, and C. Yang, "PathRAG: Pruning Graph-Based Retrieval Augmented Generation with Relational Paths," arXiv:2502.14902, 2025. [Online]. Available: https://arxiv.org/abs/2502.14902

[23] Z. Guo, L. Xia, Y. Yu, T. Ao, and C. Huang, "LightRAG: Simple and Fast Retrieval-Augmented Generation," in arxiv.org/abs/2410.05779, 2025. [Online]. Available: https://arxiv.org/abs/2410.05779

[24] M. Li, S. Miao, and P. Li, "Simple Is Effective: The Roles of Graphs and Large Language Models in Knowledge-Graph-Based Retrieval-Augmented Generation," in arxiv.org: 2410.20724, 2025. [Online]. Available: https://arxiv.org/abs/2410.20724

A. Oyewale and T. Soru "LLM-Driven Ontology Construction for Enterprise Knowledge Graphs," arXiv:2602.01276, 2026. [Online]. Available: https://arxiv.org/abs/2602.01276

[25] J. Xu, Q. Zhang, Z. Zhong, S. He, C. Zhang, Q. Lin, D. Pei, P. He, D. Zhang, and Q. Zhang, "OpenRCA: Can Large Language Models Locate the Root Cause of Software Failures?", 2025. [Online]. Available: https://openreview.net/forum?id=M4qNIzQYpd

[26] F. Zampetti, S. Geremia, G. Bavota, and M. Di Penta, "CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study," in Proc. IEEE Int. Conf. on Software Maintenance and Evolution (ICSME), 2021, pp. 471–482, doi: 10.1109/ICSME52107.2021.00048. [Online]. Available: https://doi.org/10.1109/ICSME52107.2021.00048

[27] P. R. Mazrae, T. Mens, M. Golzadeh, and A. Decan, "On the Usage, Co-Usage and Migration of CI/CD Tools: A Qualitative Analysis," Empirical Software Engineering, vol. 28, no. 52, 2023, doi: 10.1007/s10664-022-10285-5. [Online]. Available: https://doi.org/10.1007/s10664-022-10285-5

BIOGRAPHIES OF AUTHORS

Manvitha Potluri is a DevOps Cloud Solutions Architect with over 11 years of experience designing and managing scalable, cloud-native infrastructure across AWS and Azure. She has led enterprise-scale platform engineering, compliance automation, and AI-driven DevOps initiatives at leading financial services organizations, most recently at Freddie Mac, where she architected a disaster recovery framework reducing mean time to recovery from over ten hours to under thirty minutes, led the enterprise-wide migration of SSL/TLS certificate management across more than 300 Amazon EKS clusters, and designed a zero-trust security model that eliminated approximately 80% of security vulnerabilities in EKS and OpenShift environments. Her technical expertise encompasses Kubernetes orchestration, CI/CD pipeline architecture, Infrastructure as Code, and AI-assisted cloud operations. She holds AWS Certified DevOps Engineer – Professional and AWS Certified AI Practitioner certifications and has been recognized as a contributor to the AWS blog on compliance-driven DevOps. Her current research interests focus on explainable AI for cloud-native operations, knowledge graph-based infrastructure reasoning, and autonomous remediation frameworks. She is affiliated with 24X7 Systems, Chantilly, Virginia, United States. Email: connectpotluri@gmail.com

ORCID: https://orcid.org/0009-0006-3209-7857