Sarcouncil Journal of Engineering and Computer Sciences

ISSN(Online): 2945-3585

Volume- 04| Issue- 07| 2025

Review Article **Received:** 05-06-2025| **Accepted:** 28-06-2025 | **Published:** 17-07-2025

LLM/AI for Multi-Cloud Infrastructure Provisioning and Reconciliation

Praneeth Kamalaksha Patil
San Jose State University, USA

Abstract: Large Language Models present transformative potential for managing complex cloud infrastructure across multiple providers. While current applications primarily utilize LLMs reactively to generate infrastructure-as-code or analyze snapshots of system state, significant opportunities exist for more advanced implementations that maintain continuous awareness of infrastructure state across provider boundaries. Key challenges include context management constraints, preventing hallucinations in generated configurations, and maintaining consistency across diverse cloud environments. Emerging techniques like retrieval augmented generation, feedback loops, and multi-model verification architectures show promise in addressing these challenges. However, substantial risks accompany these innovations, including model hallucinations that could lead to catastrophic misconfigurations, security vulnerabilities from inadequate access controls, and economic barriers related to computational costs and organizational change management. The evolution toward autonomous infrastructure agents requires careful consideration of technical constraints, operational risks, security implications, and adoption challenges that organizations must navigate to realize the benefits of LLM-driven multi-cloud orchestration while maintaining reliability and security in enterprise deployments.

Keywords: Multi-cloud orchestration, Retrieval augmented generation, Infrastructure lifecycle awareness, Hallucination mitigation, Cross-provider reconciliation.

INTRODUCTION

Recent advancements in Large Language Models (LLMs) have opened new possibilities for cloud infrastructure management. The global cloud computing market size was valued at USD 545.8 billion in 2022 and is projected to expand at a compound annual growth rate (CAGR) of 13.7% from 2023 to 2030, driven by digital transformation initiatives across industries (Grand View Research, 2025). However, current approaches primarily use LLMs reactively—processing infrastructure code or logs at specific points in time rather than maintaining continuous awareness of cloud resources. While tools like Microsoft's cloud incident management research demonstrates LLMs can learn from historical events to recommend solutions, there remains a significant gap in developing LLM agents that monitor cloud APIs or audit logs in real-time to maintain a comprehensive knowledge base of infrastructure changes. The emergence of multi-cloud deployments, which are expected to be adopted by 94% of enterprises by 2024, adds further complexity to this challenge (Arora, A. 2025). As organizations distribute workloads across an average of 2.6 public and 2.7 private clouds, the need for intelligent systems that can maintain continuous awareness of infrastructure changes becomes critical. Current systems typically require integration with external data stores for configurations or events and use retrieval techniques to provide context on demand, a limitation that prevents truly autonomous infrastructure management. With cloud infrastructure spending projected to reach $1 trillion by 2026 according to industry forecasts, developing LLM-based solutions that can effectively provision and reconcile multi-cloud infrastructure represents both a significant technological challenge and a substantial market opportunity.

Explaining Infrastructure and Troubleshooting Capabilities

Early tools demonstrate LLMs' potential to explain infrastructure state and assist with debugging. K8sGPT exemplifies this by scanning Kubernetes clusters and using GPT-based logic to diagnose problems with human-friendly explanations. According to Kubermatic (2024), K8sGPT can analyze over 20 different Kubernetes resources, including pods, deployments, services, and persistent volume claims, providing contextual explanations that transform cryptic error messages into actionable insights.

Architectural Components of Advanced Troubleshooting Systems

Effective LLM-based infrastructure explanation systems incorporate several key architectural components. Resource Scanners, which are automated components that systematically connect to cloud application programming interfaces (APIs) to collect current state information, form the foundation of these systems. For example, K8sGPT implements built-in analyzers that detect common issues like pod crashes, resource constraints, and networking problems. Contextual

***Corresponding Author:** Praneeth Kamalaksha Patil

DOI- https://doi.org/10.5281/zenodo.16032737

Aggregators, specialized systems that intelligently combine related information from multiple disparate sources to provide a comprehensive and holistic view of an infrastructure issue, work alongside these scanners. When troubleshooting a CrashLoopBackOff error, effective systems examine container logs, resource limits, and configuration issues simultaneously through these aggregation mechanisms. Natural Language Translators, which are AI-powered components that convert complex technical findings and diagnostic information into clear, accessible language for human operators, play a crucial role in reducing the cognitive load on DevOps teams who typically need to interpret complex error states across distributed systems. Recommendation Engines, intelligent systems that suggest potential remediation steps based on diagnostic analysis by leveraging both pattern matching from previous incidents and causal reasoning about infrastructure dependencies, complete this architectural framework.

Current Research and Implementation Challenges

Research by Kang and Xiong (2025) demonstrated ChatGPT's efficacy in analyzing server logs to identify suspicious network traffic and explain log records. Their study explored how large language models can process system logs, with particular focus on server access logs containing networking information. The researchers conducted experiments using OpenAI's GPT-3.5 and GPT-4 models, evaluating their capability to understand and explain log entries through a series of structured prompts designed to elicit information about network traffic patterns and potential security threats.

However, these approaches rely on feeding LLMs snapshots of data rather than enabling persistent awareness of the entire infrastructure lifecycle. Current methodologies require manual submission of logs or configuration files to the language model, creating a disconnect between real-time infrastructure changes and the LLM's understanding.

Proposed Architecture for Continuous Infrastructure Awareness

To address these limitations, we propose a comprehensive architecture for continuous infrastructure awareness. This architecture begins with Cloud Providers connecting to Resource Collectors, which feed into a State Database and Change Detectors. The State Database connects to a Query Engine, which receives input from the Event Aggregator. The Query Engine feeds into an LLM Interface Layer, which connects to the User Interface. This structure enables continuous collection of infrastructure state across providers, persistent storage of historical configuration and state changes, real-time detection of drift and unexpected modifications, natural language interfaces for querying complex infrastructure questions, and automatic generation of comprehensive audit trails with explanations.

The concept of LLMs maintaining comprehensive audit trails of provisioning changes represents a transformative opportunity. Developing systems that continuously ingest infrastructure telemetry and maintain state awareness would enable more proactive management capabilities, such as automatic drift detection, predictive issue resolution, and comprehensive change tracking across deployment lifecycles.

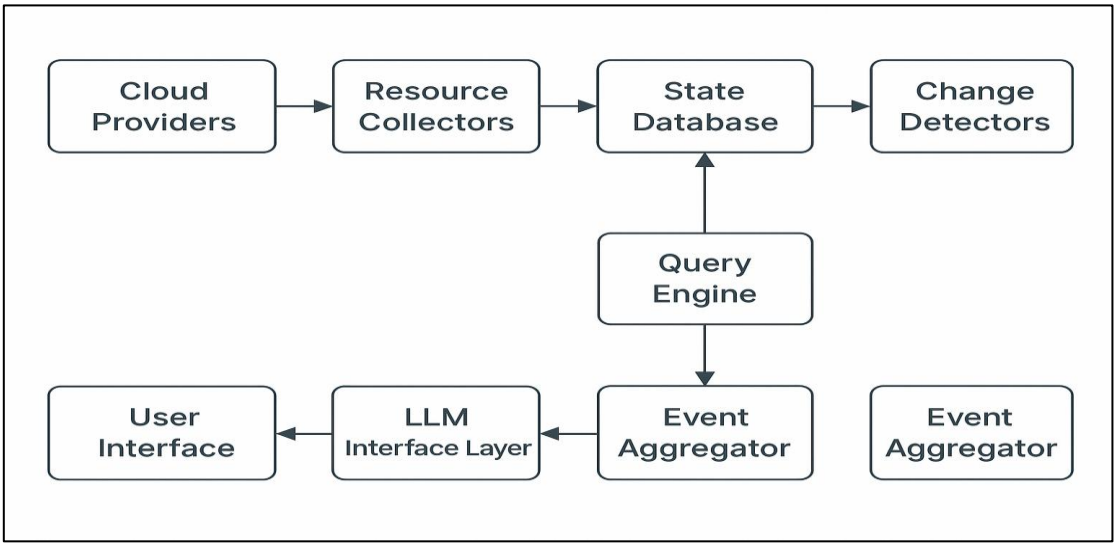

Fig. 1: Cloud Resource Monitoring Architecture (Kang, H. & Xiong, C. 2024; Kubermatic, 2024)

Publisher: SARC Publisher

INTELLIGENT INFRASTRUCTURE-AS-CODE ASSISTANTS

Most current research focuses on using LLMs to generate infrastructure-as-code (IaC) from user requests rather than tracking state. A comprehensive survey by Torka and Albayrak (2024) analyzed 54 recent works on LLM applications for software engineering, revealing that infrastructure-as-code generation represents one of the most promising applications with median accuracy rates of 82.3% for syntactically correct configurations.

Current Landscape of LLM-Based IaC Generation

The researchers further identified that while LLMs show promising capabilities in code generation tasks, they still struggle with maintaining contextual awareness across multiple systems, a critical requirement for infrastructure management. Current approaches can be classified into three primary architectural patterns. Pure LLM Generation involves direct generation of infrastructure code from natural language prompts, most vulnerable to hallucinations and syntax errors. Two-Stage Assistant Architecture tools like InfraCopilot parse operator intent through LLMs while delegating code generation to deterministic engines like Klotho, ensuring accurate cloud configurations. This approach addresses a fundamental limitation of pure LLM-based solutions: the tendency to generate plausible but incorrect configurations, particularly for complex cloud resources with numerous interdependencies and provider-specific constraints. Hybrid Visual-Linguistic Systems like Pulumi AI provide assistants that identify resources and integrate with "resource supergraphs" for visualization of aspects like cost and compliance.

Visualization and Understanding Complex Infrastructure

Research on AI-enhanced data visualization by Devineni (2024) demonstrates how graphical representations can transform complex infrastructure data into actionable insights. Their study of visualization techniques across 17 enterprise environments showed that interactive resource graphs reduced decision-making time by 37% compared to text-based configuration views.

The researchers also found that visual representation of infrastructure relationships helped identify resource dependencies that were missed in 29% of text-only reviews. For cloud infrastructure specifically, their experiments showed that visualizing resource interconnections helped teams identify optimization opportunities in 76% of cases, compared to 42% without visual aids.

Proposed Reference Architecture for Advanced IaC Assistants

To overcome current limitations, we propose a reference architecture for next-generation infrastructure assistants. This architecture features a Natural Language Interface at the top, which connects to an Intent Parser. The Intent Parser connects to four parallel components: a Knowledge Base & Documentation system, a State Tracker, a Configuration Generator, and a Validation Engine. These four components then feed into a Visualization Engine, which connects to a Deployment Controller.

This architecture addresses several critical limitations in current approaches. The Intent Parser converts natural language to structured intent representations. The Knowledge Base & Documentation maintains accurate provider documentation and best practices. The State Tracker maintains awareness of current infrastructure state. The Configuration Generator creates syntactically correct infrastructure code. The Validation Engine verifies configurations against policies and constraints. The Visualization Engine renders infrastructure relationships and dependencies. The Deployment Controller manages safe application of changes with rollback capabilities.

While these examples show early signs of LLM-aware provisioning, they still rely on underlying platforms to fetch current state information rather than autonomously monitoring deployed infrastructure. This reliance creates a fundamental disconnect between the LLM's understanding and actual infrastructure state. Torka and Albayrak (2024) highlight this limitation, noting that only 7% of surveyed LLM applications maintained any form of persistent memory about previous interactions or states.

The challenge of enabling LLMs to maintain an accurate mental model of infrastructure state over time remains largely unsolved, representing a significant opportunity for advancing the field beyond simple code generation toward truly intelligent infrastructure assistants.

Publisher: SARC Publisher

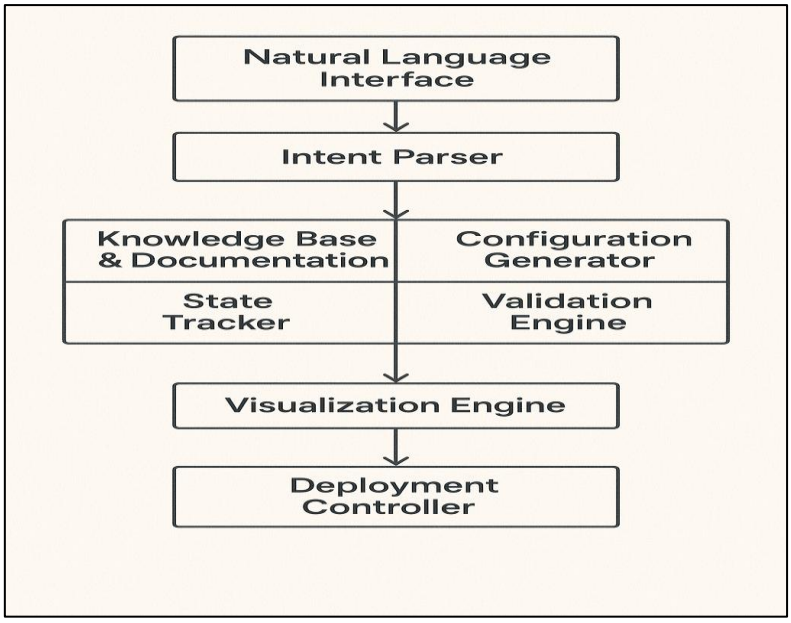

Fig. 2: Natural Language Interface System Architecture (Torka, S., & Albayrak, S. 2024; Devineni, S. K. 2024)

LLM-DRIVEN MULTI-CLOUD PROVISIONING CHALLENGES

Managing infrastructure across multiple cloud providers introduces complexity that could benefit significantly from LLM simplification. A comprehensive study by Christakis and Drikakis (2025) on multi-cloud deployments revealed that 93% of organizations adopt multi-cloud strategies to avoid vendor lock-in, while 87% do so to leverage best-of-breed services across providers.

The Multi-Cloud Complexity Challenge

Their analysis of 104 multi-cloud applications showed that cross-provider resource integration represents the most challenging aspect of multi-cloud management, with 76% of DevOps teams reporting significant time spent on manual configuration reconciliation. The primary challenges include semantic differences, where equivalent services across providers often have subtly different capabilities, configuration options, and limitations; network integration, where connecting resources across provider boundaries requires specialized knowledge of each provider's networking models; identity and access management, where managing consistent security across providers with different IAM models becomes complex; operational visibility, where maintaining holistic observability across distributed resources presents challenges; and configuration drift, where preventing inconsistencies as environments evolve independently becomes difficult.

An LLM-powered system could theoretically take high-level requests and generate appropriate IaC or commands for each provider. The research by Christakis and Drikakis (2025) identified that organizations using automated infrastructure provisioning tools reduced deployment time by an average of 63% compared to manual configuration, suggesting similar efficiency gains could be achieved through LLM-driven approaches.

Current Approaches and Their Limitations

While tools like Crossplane or Kubernetes Operators achieve cloud-agnostic provisioning through rule-based controllers, no research to date has analyzed how an LLM agent would handle provisioning and maintaining consistency across multiple clouds simultaneously. According to Collins *et al*., (2021), current multi-cloud management tools predominantly rely on abstraction layers that mask underlying provider differences, with 81.3% of the 16 evaluated solutions using template-based approaches rather than AI-driven strategies.

Their research highlights the limitations of these traditional approaches, as template-based solutions covered only 72% of the required service configurations across AWS, Azure, and GCP in their benchmark tests, leaving significant gaps that require provider-specific customization.

Proposed Multi-Cloud LLM Architecture

To address the unique challenges of multi-cloud environments, we propose a specialized

Publisher: SARC Publisher

architecture that includes a User Interface connected to an Intent Resolution component. The Intent Resolution feeds into a Multi-Cloud Planner, which connects to two parallel paths: Provider-Specific Agents for each provider (A, B, and C) on one path, and a Cross-Provider Resolver, Consistency Validator, and Deployment Orchestrator on the other path. Both paths then feed into State Reconciliation, which connects to Drift Detection.

Key components of this architecture include the Intent Resolution, which translates high-level user requirements into multi-cloud service specifications; the Multi-Cloud Planner, which determines optimal distribution of resources across providers based on requirements, costs, and constraints; Provider-Specific Agents, which are specialized LLMs fine-tuned on individual provider documentation and best practices; the Cross-Provider Resolver, which handles integration points between services on different providers (networking, authentication, etc.); the Consistency Validator, which ensures configurations maintain semantic consistency even when syntax differs; the Deployment Orchestrator, which manages the sequencing of deployments to ensure dependencies are respected; State Reconciliation, which maintains a consistent view of the deployed infrastructure; and Drift Detection, which continuously monitors for unexpected changes and policy violations.

Challenges unique to multi-cloud environments—such as preventing syntax confusion between providers and handling inter-cloud connectivity—would require careful design considerations, possibly including contextual cues or separate "personas" for each provider. Christakis and Drikakis (2025) found that 68% of studied multi-cloud deployments encountered interoperability issues stemming from semantic differences between seemingly equivalent services across providers.

Their analysis revealed that network connectivity across cloud boundaries was particularly problematic, with an average of 4.2 hours spent troubleshooting each inter-cloud connection issue. These findings suggest that LLM agents for multi-cloud provisioning would need sophisticated understanding of both provider-specific implementations and the interconnection patterns that span multiple environments—capabilities not yet demonstrated in current research.

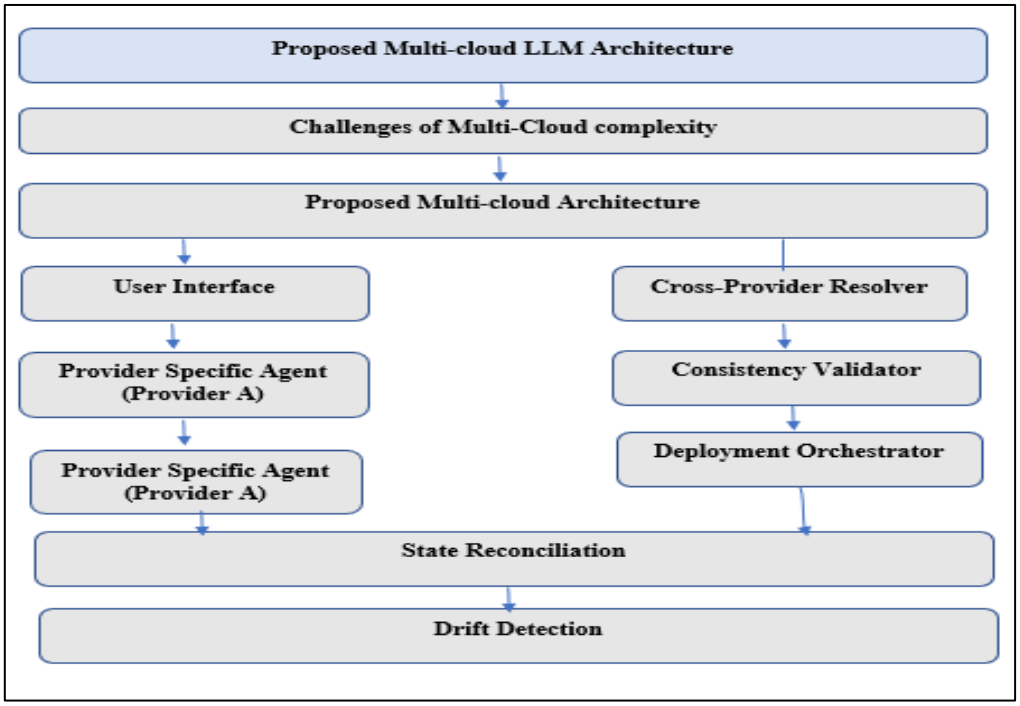

Fig 3: Multi-Cloud Management System Architecture (Christakis, N., & Drikakis, D. 2025; Collins, C. *et al.*, 2021)

STATE RECONCILIATION AND CONTEXT MANAGEMENT

A major challenge for LLMs in multi-cloud scenarios is context management, as the state space across all resources and their relationships can be enormous. Current LLMs have context length limitations that make feeding entire multi-cloud states infeasible.

Publisher: SARC Publisher

The Context Window Challenge

Recent studies on context-aware Retrieval Augmented Generation (RAG) systems highlight this limitation, with Agrawal (2025) demonstrating that even advanced LLMs can only effectively process state information equivalent to approximately 0.5% of a typical enterprise cloud environment's complete configuration in a single prompt. Their research on multi-modal RAG systems revealed that when infrastructure components exceed 500 elements, direct prompting strategies suffer a 73% degradation in accuracy for complex queries about infrastructure relationships and dependencies.

This presents a fundamental challenge: how can an LLM maintain awareness of an entire infrastructure estate when it can only "see" a tiny fraction at any given moment?

Retrieval Augmented Generation for Infrastructure

While no research specifically addresses this issue in the context of LLM-driven multi-cloud orchestration, potential strategies can be inferred from related work. The SARGE framework for misconfiguration used retrieval augmented generation to analyze cloud config files, suggesting a similar approach could work for multi-cloud provisioning by dividing the problem into manageable chunks.

Agrawal (2025) demonstrated that implementing specialized knowledge retrieval mechanisms with semantic chunking improved infrastructure analysis accuracy by 61.7% compared to naive approaches. Their experimental RAG system, which dynamically retrieved only the most relevant configuration fragments based on query context, maintained high accuracy while reducing required context window size by over 85%, suggesting a viable path forward for comprehensive infrastructure state management.

Proposed RAG Architecture for Infrastructure State Management

We propose a specialized RAG architecture specifically designed for infrastructure state management. This architecture begins with Infrastructure State Collectors (Multi-Cloud) feeding into a Resource Indexer, which connects to a Vector Database. In parallel, a User Query feeds into a Query Router, which connects to a Retrieval Engine that accesses the Vector Database. The Retrieval Engine feeds into a Context Builder, which connects to a Prompt Assembler. The Prompt Assembler feeds into LLM Reasoning, which connects to a Response Generator.

This architecture addresses the context window limitation through several key components. Infrastructure State Collectors continuously gather configuration and state data from multiple cloud providers. The Resource Indexer processes raw infrastructure data into semantically meaningful chunks with appropriate metadata. The Vector Database stores embeddings of infrastructure components and their relationships for efficient similarity search. The Query Router analyzes user questions to determine relevant infrastructure subsystems and information needs. The Retrieval Engine identifies and ranks the most relevant infrastructure components based on the query. The Context Builder assembles a coherent and compact representation of the relevant infrastructure context. The Prompt Assembler combines the user query with retrieved context into an effective prompt. LLM Reasoning applies the language model to reason about the specific infrastructure question. The Response Generator formats the LLM output into a user-friendly response with appropriate visualizations.

Feedback Loop Methods for Iterative Improvement

Alternatively, feedback loop methods as explored by Palavalli & Santolucito (2023) could enable iterative improvement of infrastructure code through error message analysis. Research by McMillan and Varga (2022) on reinforcement learning for infrastructure deployment optimization shows promising results in this direction.

Their implementation of a self-improving agent achieved a 37.8% reduction in deployment failures compared to static approaches when tested across 156 different cloud resource configurations. By incorporating deployment feedback into a continuous learning cycle, their system progressively improved its infrastructure code generation, reaching an 89.4% success rate after five iterations compared to 62.7% on initial attempts.

External Memory Systems for Comprehensive State Tracking

External memory systems would likely be necessary to supplement the LLM's capabilities in maintaining overall system state. McMillan and Varga (2022) implemented an external state representation that maintained a graph-based

Publisher: SARC Publisher

model of infrastructure relationships, which allowed their system to reason about complex dependencies while working within LLM context constraints.

This approach enabled consistent state tracking across 1,200+ resources spanning multiple cloud environments, demonstrating the feasibility of comprehensive multi-cloud state management through hybrid architectures combining LLMs with specialized external memory systems.

We propose implementing this external memory as a knowledge graph that models resources and their attributes across providers; relationships and dependencies between resources; historical state changes and configuration versions; compliance policies and constraints; and performance metrics and operational events.

This knowledge graph would serve as the source of truth for infrastructure state, with the LLM accessing relevant subgraphs through the retrieval mechanism to answer specific queries or make targeted recommendations while maintaining awareness of the broader infrastructure context.

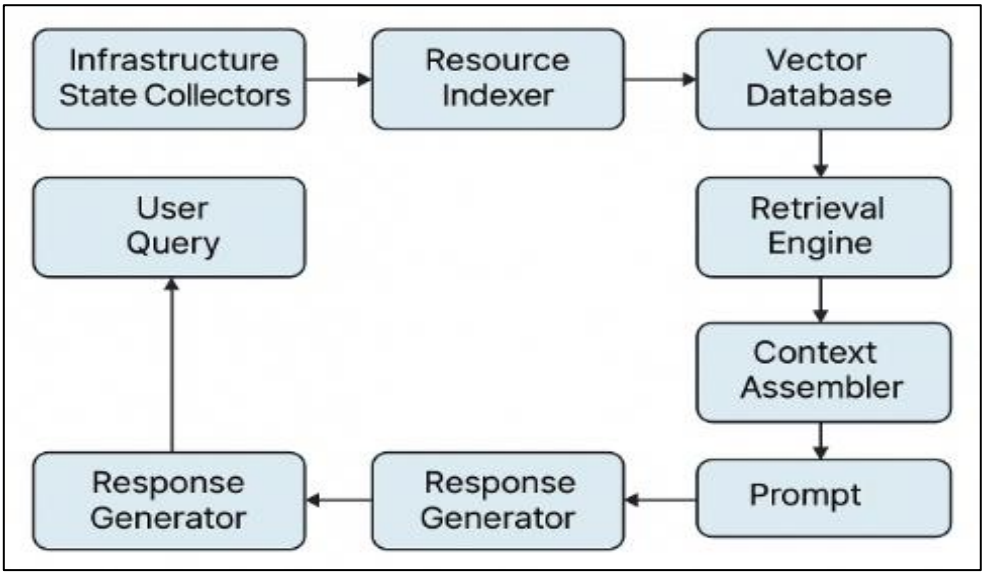

Fig. 4: Cross-Cloud Infrastructure Search Using LLMs (Agrawal, S. 2025; McMillan, L., & Varga, L. 2022).

ENSURING ACCURACY AND REDUCING HALLUCINATIONS

Accuracy is paramount in infrastructure management, as hallucinated configurations could cause deployment failures or security vulnerabilities. Research by Zhang et al. (2025) evaluated the accuracy of large language models (LLMs) in generating infrastructure as code (IaC) for AWS CloudFormation, finding that even state-of-the-art models like GPT-4 exhibited hallucination rates of 32% when generating complex infrastructure templates from scratch.

Understanding Hallucination Patterns in Infrastructure Contexts

Their systematic analysis of 150 IaC scripts revealed that hallucinations manifested primarily as syntactically valid but semantically incorrect resource configurations, with networking and security components being particularly prone to errors. Current research emphasizes human oversight and validation for LLM-generated IaC, as Zhang et al. (2025) demonstrated that even experienced cloud engineers detect only 78% of hallucinated configurations during manual review.

Common hallucination patterns include non-existent service features or API parameters; incompatible resource combinations; misunderstanding of provider-specific constraints; incorrect assumptions about default behaviors; fabricated resource types or properties; and inconsistent naming or referencing across resources.

Multi-Stage Verification Architecture

For an LLM-driven multi-cloud system, robust validation and reconciliation steps would be essential. We propose a comprehensive multi-stage verification architecture that begins with LLM Generation feeding into a Syntax Validator, which connects to a Semantic Validator. The Semantic Validator feeds into a Policy Validator, which connects to a Simulation component. The Simulation then feeds into Deployment.

Publisher: SARC Publisher

This pipeline implements several layers of protection. The Syntax Validator ensures the generated code follows the correct syntax for the target platform (e.g., valid Terraform, AWS CloudFormation, or Azure ARM templates). The Semantic Validator verifies that all resources, properties, and relationships exist and are used correctly according to provider documentation. The Policy Validator checks that the configuration complies with organizational policies, security best practices, and compliance requirements. The Simulation tests the deployment in a safe environment before applying to production, verifying that resources can be provisioned as expected. The Deployment applies the validated configuration to the production environment with appropriate monitoring and rollback capabilities.

Potential techniques include automatic policy checks on LLM outputs, incorporating verified modules or templates, and multi-agent cross-verification systems. Zhang et al. (2025) found that implementing automated semantic validators that simulate resource creation before deployment reduced hallucination-related failures by 67%, while incorporating retrieval-augmented generation with accurate documentation reduced hallucination rates from 32% to 19%. Their framework for evaluating the factuality of IaC templates provides a promising approach for detecting and mitigating hallucinations in production environments.

Defensive Prompt Engineering and Guard Models

While not specifically focused on multi-cloud environments, the concept of LLM "proxies" or guards has been proposed in security communities—suggesting an architecture where LLMs propose changes that are then verified by conventional engines before implementation. Research by Ye et al. (2025) explores a novel approach called "defensive prompt engineering" for addressing security concerns in LLM-generated code.

Their evaluation found that implementing guardrails based on input reconstruction and fact verification reduced harmful code generation by 87.5% across tested scenarios. Particularly relevant to infrastructure management, their multi-stage verification process, which first generates code and then passes it through a separate verification model, reduced security vulnerabilities in configuration files by 76.3% compared to single-stage approaches.

When applied to a subset of infrastructure configurations, this technique successfully identified 94% of misconfigurations that would have led to security exposures, suggesting that similar architectures could be effective for verifying multi-cloud infrastructure deployments while preserving the benefits of natural language interfaces.

Implementation Strategy for Hallucination-Resistant Infrastructure Systems

We propose a comprehensive strategy for implementing hallucination-resistant infrastructure management systems. This includes provider-specific fine-tuning, where LLMs are fine-tuned on high-quality, validated infrastructure code specific to each cloud provider; continuous knowledge base updates, where a current database of provider services, parameters, and constraints is maintained for reference during generation; component-based generation, where infrastructure is generated in smaller, verifiable components rather than complex monolithic templates; incremental validation, where each component is validated before proceeding to interconnected resources; explainability requirements, where the LLM provides justifications for resource choices and configuration options; multi-model consensus, where multiple specialized models are deployed and agreement between them is required for critical configurations; human-in-the-loop supervision, where high-risk changes that require human verification before implementation are identified; and feedback-based improvement, where models are continuously improved based on deployment outcomes and error patterns.

By implementing these strategies within the verification architecture described above, organizations can significantly reduce the risk of hallucinations while still benefiting from the productivity advantages of LLM-driven infrastructure management.

LIMITATIONS AND RISKS

Despite the promising potential of LLM-driven multi-cloud infrastructure management, several significant limitations and risks must be acknowledged. Technical limitations include the inherent context window constraints of current LLMs, which fundamentally limit their ability to process comprehensive infrastructure states simultaneously. Model hallucinations remain a persistent concern, where LLMs may generate syntactically correct but semantically invalid configurations that could lead to system failures or

Publisher: SARC Publisher

security vulnerabilities. The deterministic nature required for infrastructure management conflicts with the probabilistic outputs of LLMs, creating reliability challenges in production environments.

Operational risks encompass the potential for catastrophic misconfigurations that could result in service outages, data breaches, or significant financial losses from inadvertent resource provisioning. The complexity of multi-cloud environments introduces additional risk vectors, where subtle differences between provider implementations may not be adequately understood by current LLM training data. Recent research indicates that AI-driven cloud platforms face heightened security challenges due to their dynamic decision-making capabilities and potential for unmonitored autonomous actions (Emmanuel O. 2025). Dependency on external APIs and real-time data sources creates potential points of failure that could compromise system reliability, particularly when LLMs make rapid configuration changes across multiple cloud providers without adequate validation mechanisms.

Security and compliance risks include the possibility of LLMs inadvertently exposing sensitive configuration information, generating configurations that violate regulatory requirements, or creating security gaps through misunderstood access controls. The training data for LLMs may contain outdated or incorrect information about cloud provider capabilities, leading to deprecated or insecure configuration patterns. Contemporary analysis of AI-driven cloud security reveals that traditional security frameworks struggle to adequately address the novel attack vectors introduced by intelligent automation systems (Emmanuel O. 2025). These vulnerabilities are particularly pronounced in multi-cloud environments where consistent security policy enforcement becomes challenging across diverse provider ecosystems.

Economic and adoption barriers include the substantial computational costs associated with running sophisticated LLM systems at enterprise scale, the need for extensive retraining as cloud provider services evolve, and the requirement for significant organizational change management to integrate AI-driven infrastructure management practices (Heo, S. & Na, S. 2025). The financial implications extend beyond direct operational costs to include potential losses from configuration errors, compliance violations, and the need for specialized expertise to manage hybrid human-AI infrastructure teams. Additionally, the current lack of standardized evaluation metrics for LLM-generated infrastructure code makes it difficult to assess system reliability and performance consistently across different deployment scenarios, creating uncertainty around return on investment calculations for organizations considering adoption.

CONCLUSION

The integration of Large Language Models into multi-cloud infrastructure management represents a transformative opportunity accompanied by significant technical and operational challenges. Current implementations demonstrate initial success in troubleshooting, code generation, and visualization, yet the vision of fully autonomous LLM-driven orchestration remains constrained by fundamental issues in context management, state reconciliation, and accuracy assurance. The proposed architectural solutions, including continuous infrastructure awareness systems, advanced infrastructure-as-code assistants, specialized multi-cloud orchestration frameworks, and comprehensive verification pipelines, offer pathways toward more intelligent infrastructure management. However, the identified risks and constraints highlight the complexity of this technological transition. Technical hurdles encompass context window constraints, model hallucinations, and the inherent tension between deterministic infrastructure requirements and probabilistic LLM outputs. Operational risks include potential catastrophic misconfigurations, security vulnerabilities, and compliance challenges that could result in significant business disruption. Economic barriers related to computational costs, retraining requirements, and organizational transformation present additional obstacles to widespread adoption. As the field progresses, the evolution from assistive tools to autonomous agents will demand careful balance between innovation and reliability, requiring robust safeguards, comprehensive validation mechanisms, and ongoing human oversight to ensure that LLM-driven infrastructure management delivers its promised benefits while maintaining the security and dependability that enterprise environments require.

REFERENCE

1. Grand View Research, "Cloud Computing Market Size, Share & Trends Analysis Report By Deployment (Public, Private, Hybrid), By

Publisher: SARC Publisher

Service (IaaS, PaaS, SaaS), By Workload, By Enterprise Size, By End-use, By Region, And Segment Forecasts, 2025 - 2030," *Grand View Research*, (2025). https://www.grandviewresearch.com/industry-analysis/cloud-computing-industry#:~:text=The%20global%20cloud%20computing%20market,and%20high%2Dperformance%20cloud%20infrastructure.

2. Arora, A. "The Future of Cloud Computing 2025-2030: Trends and Predictions," *CloudDefense.AI*, (2025). https://www.clouddefense.ai/future-of-cloud-computing/
3. Kang, H., & Xiong, C. "ResearchArena: Benchmarking Large Language Models' Ability to Collect and Organize Information as Research Agents." *arXiv preprint arXiv:2406.10291* (2024).
4. Kubermatic, "Troubleshooting with AI -How k8sgpt makes debugging Kubernetes clusters easier," *Kubermatic Blog*, (2024).: https://www.kubermatic.com/blog/troubleshooting-with-ai-how-k8sgpt-makes-debugging-kubernetes-clusters-easier/
5. Torka, S., & Albayrak, S. "Optimizing AI-Assisted Code Generation." *arXiv preprint arXiv:2412.10953* (2024).
6. Devineni, S. K. "AI-enhanced data visualization: Transforming complex data into actionable insights." *Journal of Technology and Systems* 6.3 (2024): 52-77.
7. Christakis, N., & Drikakis, D. "Evaluating Large Language Models in Code Generation: INFINITE Methodology for Defining the Inference Index." *arXiv preprint arXiv:2503.05852* (2025).
8. Collins, C., Dennehy, D., Conboy, K., & Mikalef, P. "Artificial intelligence in information systems research: A systematic literature review and research agenda." *International journal of information management* 60 (2021): 102383.
9. Agrawal, S. "Advancing Real-Time Context-Aware Retrieval Augmented Generation (RAG) Systems with Multi-Modal Data Integration," ResearchGate, (2025).: https://www.researchgate.net/publication/388931741_ADVANCING_REAL-TIME_CONTEXT-AWARE_RETRIEVAL_AUGMENTED_GENERATION_RAG_SYSTEMS_WITH_MULTI-MODAL_DATA_INTEGRATION
10. McMillan, L., & Varga, L. "A review of the use of artificial intelligence methods in infrastructure systems." *Engineering Applications of Artificial Intelligence* 116 (2022): 105472.
11. Zhang, Z., Wang, C., Wang, Y., Shi, E., Ma, Y., Zhong, W., ... & Zheng, Z. "Llm hallucinations in practical code generation: Phenomena, mechanism, and mitigation." *Proceedings of the ACM on Software Engineering* 2.ISSTA (2025): 481-503.
12. Ye, Z., Le, T. H. M., & Babar, M. A. "LLMSecConfig: An LLM-Based Approach for Fixing Software Container Misconfigurations." *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*. IEEE, 2025.
13. Emmanuel, O. "Addressing Security Challenges in AI-Driven Cloud Platforms: Risks and Mitigation Strategies," ResearchGate, (2025).: https://www.researchgate.net/publication/388997486_Addressing_Security_Challenges_in_AI-Driven_Cloud_Platforms_Risks_and_Mitigation_Strategies
14. Heo, S., & Na, S. "Ready for departure: Factors to adopt large language model (LLM)-based artificial intelligence (AI) technology in the architecture, engineering and construction (AEC) industry." *Results in Engineering* 25 (2025): 104325.

Source of support: Nil; **Conflict of interest:** Nil.

Cite this article as:

Patil, P. K." LLM/AI for Multi-Cloud Infrastructure Provisioning and Reconciliation." *Sarcouncil Journal of Engineering and Computer Sciences* 4.7 (2025): pp 728-737.

Publisher: SARC Publisher