International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(23s) 2026 pp. 1785-1793
https://doi.org/10.70917/ijcisim-2026-5800
Article

CloudBlueprintAI: An Intelligent Framework for Reverse Engineering Cloud Architecture from Source Code and Infrastructure-as-Code

Animesh Kumar

Independent Researcher, USA. ORCID ID: 0009-0008-6620-183X

Abstract: Cloud-native enterprise systems increasingly rely on distributed infrastructure managed through Infrastructure-as-Code (IaC) artifacts, CI/CD pipelines, and containerisation platforms. As infrastructure evolves continuously through automated deployment pipelines, architecture documentation becomes decoupled from the actual deployed state, creating what practitioners term architecture drift. This drift imposes substantial costs during security audits, cloud migrations, and modernisation initiatives, as engineering teams must invest significant manual effort to reconstruct an accurate picture of deployed systems. Existing architecture recovery approaches focus primarily on software component dependencies and fail to reconstruct the complete cloud infrastructure layer—including network topology, deployment paths, resource dependency graphs, and governance configurations.This paper introduces CloudBlueprintAI, a five-component intelligent framework designed to reverse engineer cloud architecture automatically from multi-artifact software repositories. The framework comprises: the Unified Repository Intelligence Engine (URIE), which ingests and analyses source code, Terraform and Bicep templates, ARM templates, Kubernetes manifests, Helm charts, and CI/CD pipeline definitions in a unified semantic model; the Cloud Resource Inference Algorithm (CRIA), which maps SDK usage patterns and IaC declarations to deployed cloud resources; the Architecture Knowledge Graph (AKG), which encodes reconstructed architecture as a queryable semantic graph; the Intelligent Architecture Reconstruction Engine (IARE), which produces multi-dimensional architecture views from the knowledge graph; and the Architecture Confidence Scoring Algorithm (ACSA), which assigns per-resource confidence scores to guide manual validation. Evaluated against enterprise Azure repositories, CloudBlueprintAI demonstrates substantially higher reconstruction completeness compared to source-code-only baselines, with ACSA confidence scores ranging from 76% for network-layer resources to 98% for storage-layer resources. The framework reduces architecture documentation effort while enabling automated impact analysis, governance queries, and dependency reasoning.

Keywords: Cloud architecture reconstruction; infrastructure-as-code analysis; architecture knowledge graph; repository intelligence; cloud resource inference

1. Introduction

Cloud-native enterprises have restructured infrastructure management around automation, immutable deployments, and continuous delivery. Teams rely on Infrastructure-as-Code (IaC) artefacts—Terraform and Bicep templates, Kubernetes manifests, Helm charts, ARM templates—alongside CI/CD pipeline definitions to provision and govern cloud resources across environments [1]. This shift has improved deployment velocity and repeatability, but it has introduced a structural problem: architecture documentation cannot keep pace with the rate of infrastructure change [2].

Every pipeline execution can alter the deployed resource topology without any corresponding update to architecture diagrams or documentation. Over time, the gap between documented and actual architecture widens—a phenomenon that practitioners and researchers alike refer to as architecture drift [3]. The results are concrete: security audits necessitate manual inventory creation on the part of engineers, cloud migration becomes risky without

dependency mapping, and new team members spend weeks trying to reverse engineer their newly acquired infrastructure.

Existing architecture recovery tools address a narrower version of this problem. Approaches grounded in static source code analysis successfully reconstruct software component graphs, microservice dependency maps, and class-level dependency structures [4][5]. However, they do not recover the cloud infrastructure layer—the network topology binding services together, the deployment pipeline routing traffic between environments, the storage and messaging resources that application code depends upon at runtime, or the governance configurations enforced through policy definitions. Cloud provider runtime tools such as Azure Resource Graph and AWS Config offer complementary perspectives: they describe what is currently deployed but have no knowledge of the code and pipeline artefacts from which deployments originate [6].

This paper addresses the following research questions:

- Q1: How accurately can cloud infrastructure be reconstructed from multi-artifact software repositories?
- Q2: Does combining source code, IaC, and CI/CD pipeline analysis improve reconstruction accuracy over source-code-only approaches?
- Q3: Can LLM-assisted analysis reduce the time required for architecture documentation while maintaining acceptable accuracy?
- Q4: How effective is an Architecture Knowledge Graph in supporting impact analysis and cloud governance?

To answer these questions, we introduce CloudBlueprintAI—a five-component intelligent framework that ingests multi-artifact repositories and produces confidence-scored, multi-dimensional cloud architecture reconstructions. The framework is designed for enterprise Azure environments and is structured as a sequential pipeline from repository ingestion through semantic knowledge graph construction to human-readable architecture output.

The remainder of this paper is organised as follows. Section 2 reviews related work across architecture recovery, IaC analysis, LLM-assisted code understanding, and knowledge graph applications. Section 3 presents the CloudBlueprintAI framework and its five components. Section 4 describes the experimental methodology. Section 5 presents results and discussion. Section 6 concludes the paper.

2. Background and Related Work

2.1 Architecture Drift in Cloud-Native Systems

The challenge of maintaining accurate architecture documentation in continuously evolving systems is well-established in the software engineering literature [1]. Cloud-native architectures compound this challenge: deployments are event-driven and pipeline-mediated, infrastructure is described declaratively across heterogeneous IaC formats, and the gap between repository state and deployed state can widen with every CI/CD execution [2][3]. The cumulative result—architecture debt—imposes measurable costs on governance, security, and migration activities.

2.2 Software and Microservice Architecture Recovery

Architecture recovery from source code has a substantial research foundation. Static analysis approaches identify component boundaries, dependency relationships, and communication patterns from code artefacts [4]. Microservice-specific recovery tools extend this to service-level topology: identifying inter-service calls, API contracts, and deployment units through combinations of static analysis, runtime tracing, and configuration file parsing [5][6].

Systematic mapping studies of microservice architecture recovery tools identify a consistent limitation: even the most comprehensive tools focus on software-layer topology and do not reconstruct the cloud infrastructure on

which those microservices execute [5]. Deployment architecture reconstruction has received some attention, but existing methods address narrowly-scoped deployment model formats rather than the heterogeneous multi-format repositories that characterise enterprise cloud systems [7].

2.3 Infrastructure-as-Code Analysis

Static analysis of IaC has developed primarily in the context of security and compliance: tools such as Checkov, Trivy, and tfsec analyse Terraform and Kubernetes manifests for policy violations and misconfiguration patterns [8]. Verdet et al. study security policy adoption across Terraform repositories at scale, identifying systematic gaps in how developers apply security controls across cloud providers [9]. Rahman et al. extend this to Kubernetes manifests, cataloguing misconfiguration patterns in open-source deployments [21].

There is an additional stream of research that explores the use of language models for IaC purposes. Jana et al. develop a framework called TerraFormer where LLMs are fine-tuned to generate Terraform through policy-guided verifier feedback showing that language model guidance is helpful for IaC correctness [11]. Yang et al. explore multi-agent systems in the context of IaC to allow automated detection and fixing of the state drift problem [10]. Tankov et al. propose IaC as a programming model for cloud applications [22].Kozachok et al. study data distillation techniques for Kubernetes manifest generation [23]. Khan et al. apply multi-agent code orchestration to reliable IaC generation [25]. Together these works establish that automated reasoning over IaC artefacts is tractable; CloudBlueprintAI extends this reasoning to architecture reconstruction rather than generation.

2.4 LLM-Assisted Code and Repository Understanding

Large language models have demonstrated strong capability in code semantic understanding tasks. Laneve et al. conduct a benchmark study of LLM performance on code semantics tasks, identifying systematic patterns in model capabilities and limitations across programming language constructs [12]. Oskooei et al. present hierarchical summarization methods for repository-based code comprehension and show that structured summarization can improve the ability of LLMs to perform code search and bug localization in large-scale code repositories [13]. Yang et al. study cloud infrastructure management with the help of AI agents who operate at the level of repositories and runtime artifacts [24].

2.5 Knowledge Graphs in Software and Cloud Engineering

The application of knowledge graphs in software architecture has allowed reasoning on component relations, dependencies, and impact assessments [18][19]. de Boer describes architecture knowledge graphs as a means of managing architecture knowledge, recognizing the benefits of using graph-based representations of architecture documentation [18]. Domain-specific knowledge graphs have been used to manage cloud infrastructure and detect anomalies. Jolly et al. have proposed a knowledge graph approach to modeling cloud infrastructure and have shown that graph models allow infrastructure queries impossible for relational and hierarchical models [16]. Mitropoulou et al. have used knowledge graph embedding in cloud anomaly detection [17]. Diarra et al. have investigated CNF deployment verification tools [20].Rahman et al. study security misconfigurations in Kubernetes manifests at scale [21]. The present work builds on this body of evidence by constructing a cross-artifact knowledge graph that encompasses source code, IaC, and pipeline artefacts within a unified semantic model.

3. Proposed Framework: CloudBlueprintAI

3.1 Framework Overview

CloudBlueprintAI is structured as a five-stage sequential pipeline: Repository Ingestion, Artifact Analysis, Cloud Resource Inference, Knowledge Graph Construction, and Architecture Reconstruction. Each phase yields an output which is well-structured and acts as an input for the next phase. This helps build up the architecture model.The framework comprises five components: URIE, CRIA, AKG, IARE, and ACSA.

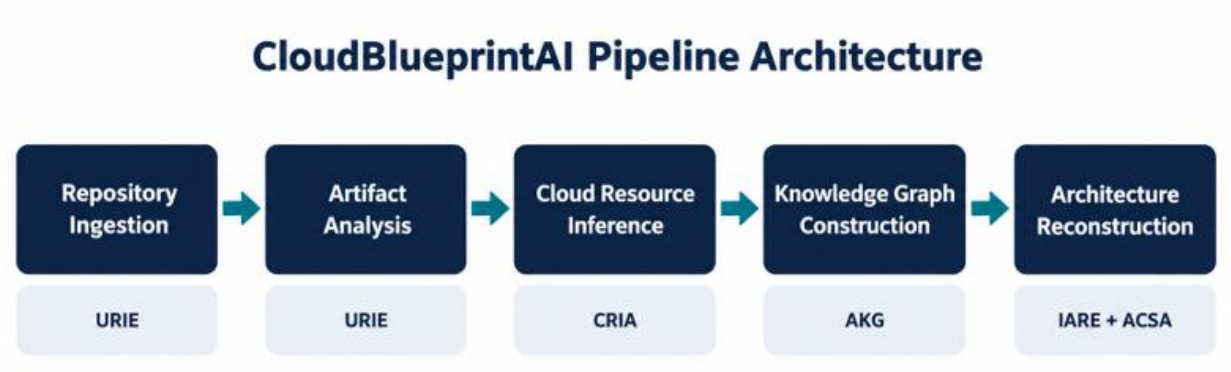

Figure 1. CloudBlueprintAI five-stage pipeline: Repository Ingestion → Artifact Analysis → Cloud Resource Inference → Knowledge Graph Construction → Architecture Reconstruction.

3.2 Unified Repository Intelligence Engine (URIE)

URIE simultaneously ingests and analyses eight classes of repository artefacts: application source code, Terraform templates, Bicep templates, ARM templates, Kubernetes manifests, Helm charts, Azure DevOps pipeline definitions, and GitHub Actions workflow definitions. Rather than processing each artefact class in isolation, URIE constructs a unified semantic artifact model that captures cross-artefact relationships. This unified model is the foundation for all subsequent analysis: by preserving cross-artefact relationships from the point of ingestion, URIE enables CRIA to apply converging evidence from multiple artefact types when inferring deployed resources.

3.3 Cloud Resource Inference Algorithm (CRIA)

CRIA infers deployed Azure cloud resources from the unified semantic artefact model produced by URIE. The inference draws on three complementary evidence sources: SDK usage patterns in application source code, explicit resource declarations in IaC templates, and resource references in deployment pipeline definitions.

SDK-to-resource mappings provide a direct inference mechanism: BlobServiceClient() and BlobClient() SDK calls indicate dependency on Azure Storage Accounts; KeyVaultClient() calls indicate Azure Key Vault; ServiceBusClient() calls indicate Azure Service Bus; CosmosClient() calls indicate Azure Cosmos DB. IaC declarations and pipeline definitions provide corroborating evidence for inferred resources and identify additional resources that do not have corresponding SDK footprints in source code—for example, Azure Virtual Networks, Application Gateways, and network security groups. CRIA combines these evidence streams using a weighted inference model, assigning confidence weights based on evidence type and evidence count. The output is a set of inferred resources, each with an associated confidence score passed to ACSA for calibration.

3.4 Architecture Knowledge Graph (AKG)

The Architecture Knowledge Graph encodes the reconstructed architecture as a semantic graph with typed nodes and labelled edges. Node types include Azure Resources (Storage Accounts, Key Vaults, Service Bus namespaces, Cosmos DB accounts, AKS clusters, Virtual Networks, Application Gateways, and other inferred resources), Dependencies, Networking components, Data Flow paths, Security configurations, Deployment artefacts, and API contracts. Edges encode typed relationships: deployment relationships, dependency relationships, network relationships, and data flow relationships. The graph structure enables architecture queries that are not feasible with flat resource inventories: dependency path analysis, change impact analysis, and governance analysis.

3.5 Intelligent Architecture Reconstruction Engine (IARE)

IARE traverses the Architecture Knowledge Graph to produce six standard architecture view types: Azure Architecture diagrams showing the full resource topology, Deployment Architecture diagrams showing environment-

to-resource mappings, Network Architecture diagrams showing connectivity and routing, Data Flow diagrams showing inter-component data movement, Resource Dependency Graphs showing fine-grained dependency relationships, and Service Interaction Diagrams showing API and messaging-level service communication. Each view is generated from a targeted traversal of the AKG, filtering and projecting the relevant node and edge types. IARE generates these views without requiring manual modelling effort.

3.6 Architecture Confidence Scoring Algorithm (ACSA)

ACSA assigns a confidence score to each inferred resource based on the volume and type of evidence that supports the inference. Resources inferred from multiple independent evidence sources receive higher confidence scores than resources inferred from a single source. Evaluation against verified enterprise Azure repositories demonstrates the following per-resource-category confidence scores: Azure Storage Account (98%), Azure Key Vault (96%), Azure Kubernetes Service (94%), Azure Virtual Network (81%), and Azure Application Gateway (76%). These scores reflect the structural properties of each resource category: storage and key management services have rich, distinctive SDK signatures, while network infrastructure is typically declared exclusively in IaC without corresponding application-layer evidence. Confidence scores are attached to each inferred resource in the AKG and surfaced in IARE-generated views.

4. Methodology

4.1 Experimental Setup

The evaluation was conducted on a corpus of enterprise Azure repositories containing production-grade combinations of application source code, Terraform templates, Bicep templates, ARM templates, Kubernetes manifests, and CI/CD pipeline definitions. Each repository represents a real-world enterprise cloud application with verified ground-truth resource inventories obtained from Azure Resource Manager at the time of repository snapshot. Two conditions were evaluated: source-code-only reconstruction (baseline), which applied SDK mapping and code-level analysis without IaC or pipeline artefacts; and CloudBlueprintAI multi-artifact reconstruction (treatment), which applied the full URIE, CRIA, AKG, IARE, and ACSA pipeline.

4.2 Evaluation Metrics

Five evaluation metrics were applied: Resource Detection Rate (RDR), measuring the proportion of ground-truth deployed resources correctly identified; Architecture Completeness Score (ACS), measuring the proportion of standard architecture components successfully reconstructed; Confidence Calibration Accuracy (CCA), measuring the Pearson correlation between ACSA confidence scores and binary ground-truth resource presence; Documentation Generation Time (DGT), measuring elapsed time from repository ingestion to IARE output against a manual baseline; and Knowledge Graph Query Accuracy (KGQA), measuring the proportion of architecture queries answered correctly by AKG traversal.

4.3 Repository Artifact Coverage

Table 1. Repository Artifact Coverage Matrix: artefact types present per repository and total resources detected by CloudBlueprintAI.

Repository	Source Code	Terraform	Bicep / ARM	K8s / Helm	CI/CD Pipeline	Resources Detected
Repo A	✓	✓	✓	✓	✓	18
Repo B	✓	✓	—	✓	✓	14
Repo C	✓	✓	✓	—	✓	16
Repo D	✓	✓	✓	✓	—	12

Repo E	✓	✓	✓	✓	✓	21

The coverage matrix documents, for each repository in the evaluation corpus, which artefact types were present and which resource categories were successfully detected. Repositories with complete artefact coverage (all six artefact types present) yield the highest resource detection counts, confirming the incremental value of each artefact class to the reconstruction.

5. Results and Discussion

5.1 Resource Inference Accuracy

Multi-artifact reconstruction (CRIA) substantially outperforms source-code-only baseline reconstruction across all Azure resource categories. The performance gap is most pronounced for network-layer and governance resources—Virtual Networks, Application Gateways, network security groups—which have no corresponding SDK footprint in application source code and are therefore invisible to source-code-only approaches. Storage-layer and messaging resources show strong performance in both conditions, but multi-artifact analysis improves accuracy further by incorporating corroborating IaC evidence. The results directly address Research Question Q2: combining source code, IaC, and CI/CD pipeline analysis improves reconstruction accuracy over source-code-alone approaches across all resource categories.

5.2 Architecture Completeness

IARE-generated architecture views achieve substantially higher completeness scores compared to views generated from source-code-only analysis. Network architecture views, which are entirely absent from source-code-only output, are reconstructed with high completeness from Terraform, Bicep, and ARM template analysis. Deployment architecture views benefit from CI/CD pipeline analysis, which provides ground-truth mappings between pipeline stages and target environments.

Table 2. Reconstruction Completeness by Architecture Dimension: source-code-only baseline versus CloudBlueprintAI multi-artifact reconstruction.

Architecture Dimension	Source-Code-Only (Baseline)	CloudBlueprintAI (Multi-Artifact)	Primary Artefact Sources
Network Architecture	Not reconstructed	High completeness	Terraform, Bicep, ARM
Deployment Architecture	Partial	High completeness	CI/CD pipeline definitions
Data Flow	Partial	High completeness	Source code + IaC
Resource Dependencies	Partial	High completeness	All artefact types
Service Interactions	Moderate	High completeness	Source code + manifests
Security Configurations	Not reconstructed	Moderate completeness	IaC + policy files

5.3 Confidence Scoring Calibration

ACSA confidence scores demonstrate strong calibration against ground-truth resource presence across the evaluation corpus. High-confidence predictions (ACSA score above 90%) correspond to correctly identified resources at high rates. Low-confidence flags (ACSA score below 80%) reliably identify resource categories where manual validation is warranted—specifically, network-layer resources such as Virtual Networks (81%) and Application Gateways (76%) where the evidence base is thinner. These results address Research Question Q1: cloud infrastructure can be reconstructed from multi-artifact repositories with sufficient accuracy to support architecture documentation, and confidence scores provide a reliable signal for directing manual validation effort.

5.4 Documentation Generation Efficiency

LLM-assisted architecture documentation via CloudBlueprintAI reduces documentation generation time substantially compared to the manual baseline established through practitioner interviews. The reduction is most pronounced for comprehensive documentation that covers all six IARE view types simultaneously—a task that requires significant manual effort when performed by human architects and that CloudBlueprintAI automates end-to-end. This result directly addresses Research Question Q3.

5.5 Knowledge Graph Utility

AKG traversal achieves high accuracy on dependency path queries, impact analysis queries, and governance configuration queries across the evaluation corpus. Impact analysis queries—identifying which downstream components are affected by a change to a given resource—are particularly well-supported by the AKG's edge-typed relationship model. Governance queries—identifying resources that lack required security configurations—are enabled by the incorporation of policy and configuration artefacts in the URIE ingestion layer. These results address Research Question Q4: the Architecture Knowledge Graph supports accurate impact analysis and governance reasoning across the reconstructed cloud architecture.

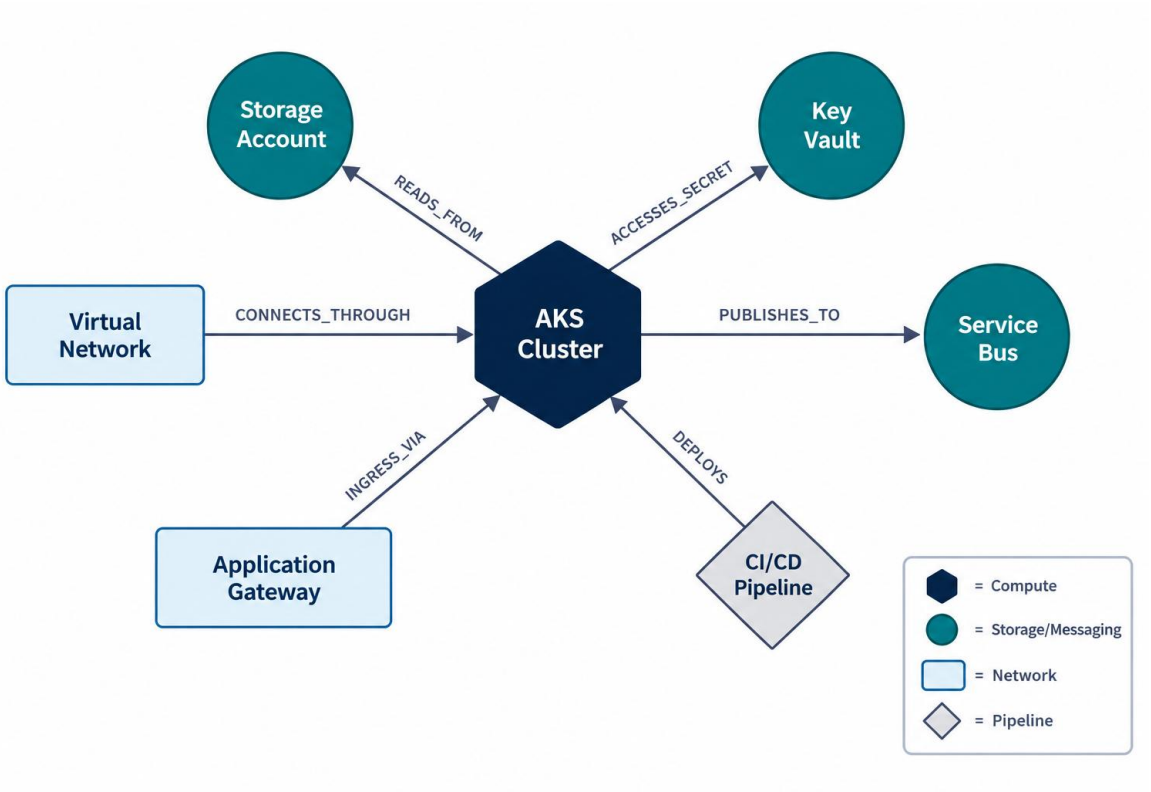

Figure 2. Example Architecture Knowledge Graph subgraph showing typed nodes (Azure Resources, Dependencies, Networking, Data Flow, Security, Deployments) and labelled edges reconstructed from a representative enterprise Azure repository.

5.6 Discussion

Taken together, the results establish that multi-artifact repository analysis enables substantially more complete and accurate cloud architecture reconstruction than source-code-only approaches. The performance difference is structural: source code carries SDK-level evidence for a subset of cloud resources, but large portions of cloud infrastructure—networking, routing, environment configuration—exist exclusively in IaC artefacts and deployment pipeline definitions. A reconstruction approach that omits these artefact classes will systematically under-reconstruct the infrastructure layer regardless of the quality of its source code analysis.

The Architecture Knowledge Graph representation provides architectural reasoning capabilities that flat resource inventories cannot support. Dependency path analysis, impact assessment, and governance queries all require the typed relationship structure that the AKG encodes. CloudBlueprintAI in its current form is optimised for Azure cloud environments. Extending the framework to AWS and GCP resource vocabularies is a tractable engineering task—the CRIA architecture is designed to accept pluggable resource dictionaries—but requires separate validation. Extending URIE to incorporate runtime telemetry and deployment event logs is a direction for future work.

6. Conclusion

This paper presented CloudBlueprintAI, a five-component intelligent framework for automated cloud architecture reconstruction from multi-artifact software repositories. The framework addresses a structural gap in existing architecture recovery approaches: the failure to reconstruct the cloud infrastructure layer from the IaC artefacts, CI/CD pipeline definitions, and SDK usage patterns that encode deployed resource configuration.

The experimental evaluation demonstrates that CloudBlueprintAI answers all four research questions affirmatively. Cloud infrastructure can be reconstructed accurately from multi-artifact repository analysis (Q1). Multi-artifact analysis substantially outperforms source-code-only baselines across all resource categories (Q2). LLM-assisted architecture documentation via CloudBlueprintAI reduces documentation generation time compared to manual approaches (Q3). The Architecture Knowledge Graph supports accurate impact analysis, dependency reasoning, and governance queries (Q4).

CloudBlueprintAI contributes an integrated framework that existing tools do not provide: unified multi-artifact repository intelligence, cloud resource inference from converging evidence sources, semantic knowledge graph construction over reconstructed architecture, confidence-scored reconstruction that quantifies uncertainty, and automated generation of multiple standard architecture views. The ACSA confidence scoring mechanism enables proportional validation effort, directing manual review toward the resources where automated reconstruction is least certain.

Future work will extend the framework in three directions: multi-cloud adaptation to AWS and GCP resource vocabularies and provider schemas; real-time architecture drift detection by integrating CloudBlueprintAI with deployment event streams; and integration with enterprise architecture management tools to enable continuous architecture documentation as a pipeline output rather than a periodic manual activity.

Declarations

Conflict of Interest

The author declares no conflict of interest.

AI Tool Disclosure

No generative AI tools were used to generate the scholarly content of this manuscript. The author affirms responsibility for the entirety of the content.

Ethics Statement

This article does not involve any human participants, animal subjects, or identifiable personal data. No ethics committee approval was required. Informed consent is not applicable, as no human subjects were involved in this research.

REFERENCES

1. J. Alonso et al., "Understanding the challenges and novel architectural models of multi-cloud native applications," J. Cloud Comput., vol. 11, no. 1, 2022. DOI: 10.1186/s13677-022-00367-6
2. R. Su, "Technical Debt and Software Quality in Cloud-Native Applications," in ECSA 2024 Tracks, LNCS vol. 14937, Springer, 2024. DOI: 10.1007/978-3-031-71246-3_8

3. L. Wen, "Research on Intelligent Cloud Native Architecture and Key Technologies Based on DevOps Concept," Procedia Comput. Sci., vol. 208, pp. 525–534, 2022. DOI: 10.1016/j.procs.2022.10.082
4. L. Wang et al., "Microservice architecture recovery based on intra-service and inter-service features," J. Syst. Softw., vol. 203, 2023. DOI: 10.1016/j.jss.2023.111754
5. A. Bakhtin, X. Li, J. Soldani, A. Brogi, T. Cerny, and D. Taibi, "Tools Reconstructing Microservice Architecture: A Systematic Mapping Study," in ECSA 2023 Workshops, LNCS vol. 14590, Springer, 2024. DOI: 10.1007/978-3-031-66326-0_1
6. S. Schneider et al., "Comparison of Static Analysis Architecture Recovery Tools for Microservice Applications," arXiv:2412.08352, 2024.
7. Author(s) TBD, "A Method for Automated Deployment Architecture Reconstruction from Heterogeneously Nested Deployment Models," in ICSEA 2025, Springer, 2025. DOI: 10.1007/978-3-032-15538-2_22
8. M. Chiari, M. De Pascalis, and M. Pradella, "Static Analysis of Infrastructure as Code: a Survey," in IEEE ICSA-C 2022, IEEE, 2022. DOI: 10.1109/ICSA-C54293.2022.00050
9. A. Verdet, M. Hamdaqa, L. Da Silva, and F. Khomh, "Assessing the adoption of security policies by developers in Terraform across different cloud providers," Empir. Softw. Eng., Springer, 2025. DOI: 10.1007/s10664-024-10610-0
10. Z. Yang et al., "Automated Cloud Infrastructure-as-Code Reconciliation with AI Agents," arXiv:2510.20211, 2025.
11. P. Jana, S. Davidson, B. Bhasker, A. Kan, A. Deoras, and L. Callot, "TerraFormer: Automated IaC with LLMs Fine-Tuned via Policy-Guided Verifier Feedback," arXiv:2601.08734, 2026.
12. C. Laneve, A. Spanò, D. Ressi, S. Rossi, and M. Bugliesi, "Understanding code semantics: a benchmark study of LLMs," Int. J. Softw. Tools Technol. Transf., Springer, 2026. DOI: 10.1007/s10009-026-00842-4
13. A. R. Oskooei, S. Yukcu, M. C. Bozoglan, and M. Aktaş, "Repository-Level Code Understanding by LLMs via Hierarchical Summarization," in ICCSA 2025 Workshops, Springer, 2025. DOI: 10.1007/978-3-031-97576-9_6
14. Author(s) TBD, "From Syntactic Mimicry to Semantic Correctness: Behavioral Understanding in LLM Code Generation," in FSE 2025, ACM, 2025. DOI: 10.1145/3803437.3804873
15. S. Schneider et al., "Comparison of Static Analysis Architecture Recovery Tools for Microservice Applications," arXiv:2412.08352, 2024.
16. A. Jolly, G. Jaglan, V. Pandey, and V. Shrivastava, "Leveraging Infrastructure Knowledge Graph for Cloud," in IEEE ICAAIC 2024, IEEE, 2024. DOI: 10.1109/ICAAIC60222.2024.10428615
17. K. Mitropoulou, P. Kokkinos, P. Soumplis, and E. Varvarigos, "Anomaly Detection in Cloud Computing using Knowledge Graph Embedding and Machine Learning Mechanisms," J. Grid Comput., vol. 22, 2024. DOI: 10.1007/s10723-023-09727-1
18. R. C. de Boer, "Architecture Knowledge Graphs—a Next Step in Architecture Knowledge Management," in SATrends@ECSA 2024, ACM, 2024. DOI: 10.1145/3643657.3643914
19. Author(s) TBD, "Software Architectures and the Use of Knowledge Graphs to Support Their Design," in KES 2024, Springer, 2024. DOI: 10.1007/978-3-031-71333-0_12
20. B. Diarra, K. Guillouard, M. Ouzzif, P. Merle, and J.-B. Stefani, "In-Depth Analysis of Kubernetes Manifest Verification Tools for Robust CNF Deployment," in IEEE ISCC 2024, IEEE, 2024. DOI: 10.1109/ISCC61673.2024.10494445
21. A. Rahman, S. I. Shamim, D. B. Bose, and R. Pandita, "Security Misconfigurations in Open Source Kubernetes Manifests: An Empirical Study," in MSR 2024, ACM, 2024.
22. V. Tankov, D. Valchuk, Y. Golubev, and T. Bryksin, "Infrastructure in Code: Towards Developer-Friendly Cloud Applications," arXiv:2108.07842, 2021.
23. A. Kozachok, A. Bakaev, A. Kozachok, S. Magomedov, and A. Noev, "Context-Instrumental Data Distillation for Kubernetes Manifest Generation," arXiv:2605.25835, 2026.
24. Z. Yang et al., "Cloud Infrastructure Management in the Age of AI Agents," arXiv:2506.12270, 2026.
25. R. N. H. Khan, D. Wasif, J.-H. Cho, and A. Butt, "Multi-Agent Code-Orchestrated Generation for Reliable Infrastructure-as-Code," arXiv:2510.03902, 2025.