IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

Open Challenges in Automated Deployment from GitHub to AWS: A Literature Review

H. B. Magar, Devika Mule, Sai Nimbalkar, Madiha Shaikh
Information Technology Department
AISSMS IOIT, Pune, India
harshada.magar@aissmsioit.org, devika.mule@aissmsioit.org
sai.nimbalkar@aissmsioit.org, madiha.shaikh@aissmsioit.org

Abstract: *Automating the path from a source repository to running cloud infrastructure remains a manual, error-prone task for many development teams. Recent work applies large language models (LLMs) and autonomous agents to this problem, spanning GitHub workflow generation, Infrastructure-as-Code (IaC) synthesis, drift reconciliation, and policy-bounded deployment decisions on cloud platforms such as Amazon Web Services (AWS). This paper reviews 30 recent publications at the intersection of artificial intelligence, GitHub-based automation, AWS, and automated deployment, with an in-depth analysis of eight core studies. We compare their architectures, AI techniques, evaluation methods, and reported outcomes, and organize the corpus into five themes. The review finds that LLM-based IaC generation reaches high deployment success mainly through iterative feedback, that generated infrastructure tends to over-provision relative to expert baselines, and that rigorous, comparable evaluation on realistic AWS workloads remains scarce. We conclude with open challenges and a conceptual roadmap for future work. The paper is literature-based and presents no system implementation.*

Keywords: artificial intelligence, large language models, GitHub Actions, AWS, CI/CD, Infrastructure as Code, automated deployment, agentic AI, DevOps, literature review.

I. INTRODUCTION

Moving an application from a source repository to running cloud infrastructure requires developers to extract application requirements, translate them into infrastructure specifications, configure services, provision resources, deploy code, and validate the result. These steps demand specialized expertise and remain largely manual and error-prone [1]. GitHub has become the default home for this workflow: GitHub Actions was adopted by roughly 30% of the 5,000 most popular repositories examined by Wessel et al. [2], and a wider ecosystem of bots and workflow automation has grown around it [3]. On the infrastructure side, Amazon Web Services (AWS) provides managed pipeline services such as CodePipeline, CodeBuild, and CodeDeploy [4], and declarative templating through AWS CloudFormation [5].

Large language models (LLMs) and autonomous agents are now being applied across this chain. Studies have used LLMs to generate GitHub Actions workflows [6], to synthesize deployable IaC templates [5], to reconcile infrastructure drift [7], to configure and deploy cloud systems [8], and to make policy-bounded release decisions inside CI/CD pipelines [9]. A recent systematic review of AI for IaC finds that LLMs dominate generation tasks while classical machine learning dominates analysis tasks, and that operations phases are less studied than development phases [10].

These advances are particularly relevant to small teams building new web platforms, where dedicated DevOps expertise is limited and end-to-end deployment automation from GitHub to AWS could substantially reduce effort and risk. However, the evidence is scattered across venues, methods, and maturity levels, which makes it hard to judge what is feasible today.

This paper therefore reviews the literature at the intersection of four themes: artificial intelligence, GitHub-based automation, AWS, and automated deployment. Its contributions are: (i) a curated corpus of 30 recent publications; (ii) an in-depth critical review of eight core studies; (iii) a comparative synthesis that groups the corpus into thematic clusters;

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

and (iv) a set of open challenges and research directions. The paper is a literature-based study; it does not present or evaluate a system implementation. Section II gives background, Section III describes the review method, Section IV reviews eight core papers, Section V compares and synthesizes the corpus, Section VI discusses challenges and future directions, and Section VII concludes.

II. BACKGROUND

A. GitHub-Centered CI/CD Automation

GitHub Actions lets repositories define automated workflows triggered by repository events. Wessel et al. report that adoption was associated with more rejected pull requests and a longer time to accept a pull request [2], showing that automation changes collaboration dynamics and is not uniformly beneficial. At scale, Chomątek et al. used DeepSeek-V2 and GPT-4o to analyze pipeline definition files from 28,770 popular GitHub repositories and found near-universal build (80.0%) and test (80.2%) stages but only 12.1% adoption of static application security testing [11]. Calefato et al. observed that MLOps workflows built on GitHub Actions and CML were still uncommon in open-source projects [12], while Garg et al. discuss CI/CD for automated deployment of machine learning models [13]. AI also enters at review time: Cihan et al. study AI-assisted code-review tools, including GitHub Copilot, in an industrial setting [14].

B. AWS-Based Deployment and IaC

On AWS, deployment automation spans pipelines, containers, and scaling. Cepuc et al. describe a CI/CD pipeline for containerized applications on AWS using Jenkins, Ansible, and Kubernetes [15]. Naayini proposes a Kubernetes-based architecture for AI applications that combines containerization, orchestration, and continuous delivery with MLOps [16]. Machine learning has also been applied to resource management: Zhang et al. combine load forecasting with reactive scaling on EC2 and report a 33.7% increase in CPU utilization and a 32.7% reduction in operational costs relative to threshold-based scaling [17], and Stojanović et al. compare AWS architectures for scalable, cost-efficient retrieval-augmented generation [18]. IaC connects pipelines to infrastructure: Karanam studies Git-driven IaC versioning and rollback with Terraform across AWS, Azure, and GCP [19].

C. AI and Agents in DevOps

A growing body of work positions LLM-based agents as active participants in software delivery. Nagvekar frames agentic AI as systems able to perceive, reason, and act within CI/CD [20], and Barnes et al. argue that the central design question is authority transfer: how much operational decision authority is delegated to agents and where control resides [21]. Framework-style proposals include ML-based deployment risk prediction, rollback management, and monitoring [22]; LLM-plus-reinforcement-learning optimization of autonomous pipelines [23]; AI-driven DevOps for multi-cloud deployment [24]; ML-driven CI/CD monitoring and resource optimization for Java systems [25]; and a master's thesis on agentic deployment [26]. Related work addresses automated software environment setup [27], detection of malicious IaC changes [28], and LLM-based security policy validation [29].

III. REVIEW METHODOLOGY

A. Scope and Selection

The review covers publications from 2020 to 2026 that address at least one of four themes: (T1) AI, machine learning, LLMs, or agents; (T2) GitHub or repository-based workflows; (T3) AWS or comparable cloud infrastructure; and (T4) automated deployment, including CI/CD and IaC. Starting from a working collection of roughly 200 unique papers, the full text of each document was scanned for theme-specific terms, and the highest-scoring candidates were screened by reading their abstracts. Thirty papers were retained. Eight were chosen for in-depth review because they most directly address automated deployment on cloud platforms with GitHub or AWS relevance; seven of them use AI, and one AWS-native pipeline is included as a non-AI baseline.

B. Classification and Limitations

Papers were coded by layer (pipeline, IaC, configuration, operations), AI technique, platform, and type of evidence. This is a narrative review, not a PRISMA-style systematic review. The corpus mixes peer-reviewed conference and journal

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

papers, arXiv preprints, and a master's thesis. Reported figures are author-reported and have not been independently verified here. Because the corpus is a curated selection, relevant work outside it may have been missed.

IV. REVIEW OF EIGHT CORE STUDIES

A. CloudBot: Code-to-Cloud Deployment

CloudBot [1] targets the "code-to-cloud" problem directly. It chains three components, GitHub Analyst, Cloud Architect, and Cloud Engineer, that extract requirements from a GitHub repository, design infrastructure, and generate validated Terraform templates. LLMs enhanced with retrieval-augmented generation (RAG) map application needs to infrastructure specifications. Evaluated on AWS with text-processing, image-analysis, and video-processing use cases, all generated configurations deployed and executed correctly. However, similarity to expert-written baselines was only 18–43% across syntax, semantic, and functional dimensions, and the generated configurations tended toward over-provisioning. This is the closest match to the theme of this review, but its evidence is limited to three use cases and it establishes a baseline rather than expert-level quality.

B. Deployability-Centric IaC Generation

Zhang et al. [5] argue that IaC generation should be judged by deployability, not only syntax. They build DPIaC-Eval, a benchmark of 153 real-world scenarios, and propose IaCGen, an LLM framework that iteratively feeds deployment errors back to the model. Targeting AWS CloudFormation, they report that Claude-3.5 and Claude-3.7 achieved only 30.2% and 26.8% deployment success on the first attempt, whereas with IaCGen all evaluated models exceeded 90% passItr@25 and the two Claude models reached 98%. The work shows that feedback from real deployment attempts is the key ingredient; success after many iterations also suggests cost and latency effects that single-shot figures hide.

C. NSync: Automated IaC Reconciliation

Yang et al. [7], from the University of Michigan and AWS, address infrastructure drift, which arises when resources managed by IaC are modified through consoles, CLIs, or SDKs. NSync is an agentic system that uses cloud APIs to detect changes, cleans and consolidates noisy API traces, generates patches to the IaC configuration, and continuously learns from experience. The authors curate a dataset of 372 validated cases from five real-world Terraform projects and report 0.97 accuracy, a 26% improvement over the baseline, with 1.5× lower token usage. The paper is important for post-deployment maintenance, a phase that the IaC review finds under-studied [10]; its scope is Terraform-based reconciliation rather than first-time deployment.

D. LLM-Generated GitHub Actions Workflows

Mehta et al. [6] evaluate GPT-3.5 and GPT-4 for generating GitHub Actions workflows using public repositories, prompt engineering, and three metrics: exact match, BLEU, and a new "DevOps Aware" score. GPT-4 showed clear gains in DevOps awareness and syntactic correctness. They also release a GitHub App, built with Probot, that generates workflows and commits them to a repository. The authors list extending the scope to deployment steps as future work, so the study covers CI automation rather than cloud rollout, and its metrics measure similarity to reference workflows rather than successful execution.

E. LADs: LLM-Driven Cloud Configuration

Khan et al. [8] present LADs, an LLM-driven framework for cloud configuration and deployment. It combines retrieval-augmented generation, few-shot learning, chain-of-thought prompting, and feedback-based prompt chaining so that the system generates configurations and learns from deployment failures. The authors analyze which optimizations work under which conditions, discuss trade-offs among performance, cost, and scalability, and state that LADs reduces manual effort and improves reliability; the framework and dataset are open-sourced. Its emphasis on failure-driven refinement parallels IaCGen [5], but it focuses on configuring distributed systems rather than deploying an arbitrary repository.

F. Auto Deploy-AWS: An AWS-Native Baseline

Chauhan and Nayyer [4] implement Auto Deploy-AWS, an automated CI/CD deployment for Node.js applications that uses the Boto3 Python SDK to orchestrate AWS CodePipeline, CodeBuild, and CodeDeploy under an IaC approach. The pipeline adds IAM-based access control, encrypted artifacts, and automatic rollback on failure, with scaling through Auto

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

Scaling Groups and Elastic Load Balancing. They report a 68% reduction in deployment time (25 to 8 minutes) and a 99.8% deployment success rate. The paper uses no AI, but it documents the AWS-native building blocks that AI-driven approaches must ultimately target and offers a conventional baseline against which AI-generated pipelines can be compared.

G. AI-Augmented CI/CD with Autonomous Decisions

Baqar et al. [9] propose LLMs and autonomous agents acting first as policy-bounded copilots and then as decision makers at points such as interpreting flaky tests, choosing rollback strategies, tuning feature flags, and promoting canary releases. Contributions include a reference architecture, a decision taxonomy with a policy-as-code guardrail pattern, a trust-tier framework for staged autonomy, and an evaluation approach based on DORA metrics. In a case study migrating a React 19 microservice, they report up to 35% lower lead time and 43% lower mean time to recovery, with 87.5% accuracy of agent interventions and a 14.3% human override rate. The governance contribution is the main strength; the empirical evidence rests on a single case study.

H. Generative AI as an Infrastructure Copilot

Esposito et al. [30] present a vision for generative AI that consumes artifacts from each DevSecOps stage to generate and iteratively refine IaC scripts, positioning it as a lifecycle-aware alternative to single-prompt generation. They provide a roadmap for implementation and empirical validation but no experimental results. The value of the paper lies in its framing; its claims remain hypotheses until validated against human experts and state-of-the-art tools, which the authors themselves identify as future work.

V. COMPARATIVE ANALYSIS AND SYNTHESIS

A. Cross-Study Comparison

Table I summarizes the eight core studies along layer, AI method, platform, and headline result.

TABLE I
COMPARISON OF THE EIGHT CORE STUDIES

Ref.	Layer	AI method	Platform	Key reported result
[1]	Repo-to-cloud deployment	LLM + RAG, multi-agent	GitHub; AWS, Terraform	All 3 use cases deployed; 18–43% similarity to expert IaC
[5]	IaC generation	LLM with iterative error feedback	AWS CloudFormation	98% (Claude) after iteration vs. 30.2% / 26.8% first attempt
[7]	Drift reconciliation	LLM agents with learning	Terraform; cloud APIs	0.97 accuracy; +26% over baseline
[6]	Workflow generation	GPT-3.5 / GPT-4 prompting	GitHub Actions	GPT-4 better DevOps awareness and syntax
[8]	Cloud config. and deployment	RAG, few-shot, CoT, prompt chaining	Cloud systems	Reduced manual effort; learns from failures
[4]	AWS-native CI/CD	None (non-AI baseline)	AWS CodePipeline, CodeBuild	25 to 8 min deploy; 99.8% success
[9]	CI/CD decisions	LLM agents, policy-as-code	CI/CD; React microservice case	Up to 35% lower lead time; 43% lower MTTR
[30]	IaC across DevSecOps	Generative AI (vision)	Not applicable	Roadmap only; no results

Three observations follow. First, evidence quality varies: IaCGen, NSync, and CloudBot report quantitative evaluations against benchmarks or baselines, whereas Baqar et al. rely on one case study and Esposito et al. offer no results. Second, feedback-driven refinement recurs across studies, using deployment errors in IaCGen, deployment failures in LADs, and API traces in NSync, which suggests that closing the loop with the real environment is the most consistent lever for

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

reliability. Third, the AWS-native pipeline of Chauhan and Nayyer reports conventional operational metrics, deployment time and success rate, that the AI-based studies rarely report, but it contains no AI component.

B. Thematic Synthesis of the 30-Paper Corpus

Table II assigns each of the 30 papers to one primary theme. Repository-driven work establishes how automation is actually used on GitHub and how LLMs can generate its artifacts. IaC work concentrates on generation, reconciliation, and versioning. AWS-native work provides the target environment and cost-oriented resource management. Agentic CI/CD work addresses autonomy and governance, and the last group covers security validation, MLOps, and environment setup.

TABLE II
THEMATIC GROUPING OF THE REVIEWED PAPERS

Theme	References
Repository- and GitHub-driven automation	[1], [2], [3], [6], [11], [12], [14]
IaC generation, reconciliation, and configuration	[5], [7], [8], [10], [19], [30]
AWS-native pipelines and resource management	[4], [15], [16], [17], [18]
Agentic and AI-augmented CI/CD	[9], [20], [21], [22], [23], [24], [25], [26]
Security, MLOps, and environment setup	[13], [27], [28], [29]

VI. CHALLENGES AND FUTURE DIRECTIONS

A. Open Challenges

Evaluation and benchmarks. Evaluations differ widely in metrics and scale. CloudBot compares against expert baselines on three use cases [1], IaCGen introduces a 153-scenario benchmark [5], and NSync a 372-case dataset [7], but these are not comparable with one another. A shared benchmark covering repository analysis, IaC generation, deployment success, and cost on AWS is missing.

Cost and over-provisioning. Generated infrastructure tends toward over-provisioning [1], while ML-based scaling can reduce costs after deployment [17]. How to combine generation-time right-sizing with runtime optimization is unexplored in the reviewed corpus.

Governance and trust. Delegating release decisions to agents raises questions of authority, auditability, and human override [9], [21]. Staged trust tiers and policy-as-code guardrails are promising but are supported so far by a single case study.

Security of generated artifacts. LLMs can augment rule-based IaC scanning for Kubernetes, IAM, and Terraform [29], and semantic analysis may help detect malicious IaC changes [28]. Since only 12.1% of surveyed GitHub repositories use static application security testing [11], security validation is unlikely to be present by default.

End-to-end integration. Within the reviewed corpus, no single study connects LLM-generated GitHub workflows, IaC generation, security validation, AWS deployment, and drift reconciliation in one pipeline. Mehta et al. name deployment steps as future work [6], and CloudBot starts from GitHub analysis but provisions through Terraform rather than a GitHub workflow [1].

Operations after deployment. Operations phases are less studied than development phases [10], although drift [7] and rollback [4], [19] are central to keeping deployed systems healthy.

B. A Conceptual Reference Flow

Synthesizing the reviewed work suggests a conceptual flow with six stages: (1) a repository event triggers analysis of the codebase [1]; (2) an LLM drafts workflow and IaC artifacts [5], [6]; (3) policy and security validation checks them [28], [29]; (4) deployment to AWS proceeds with feedback from failed attempts [5], [8] and rollback support [4]; (5) monitoring and drift reconciliation maintain the deployed state [7]; and (6) human approval gates scale with a trust tier [9], [21]. This flow is offered as an organizing hypothesis for future work, not as an implemented or validated system.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

VII. CONCLUSION

This paper reviewed 30 recent publications on AI-driven automated deployment from GitHub to AWS and analyzed eight in depth. The evidence shows that LLMs can generate workflows and infrastructure code and can deploy simple applications end to end, but quality depends heavily on iterative feedback from real deployment attempts, generated infrastructure is often over-provisioned, and governance and security validation are still immature. The main gap is an integrated, rigorously evaluated pipeline that links GitHub workflows, IaC generation, validation, AWS deployment, and post-deployment maintenance. Future work should develop shared benchmarks, report cost and latency alongside success rates, and validate staged-autonomy designs in more than one case study.

REFERENCES

[1] O. Ódùnsī, A. Mohan, S. A. M. Mostafa, and J. Wang, "CloudBot: Autonomous end-to-end cloud deployment from code to infrastructure," in Proc. IEEE Int. Conf. Big Data (BigData), 2025, doi: 10.1109/BIGDATA66926.2025.11402281.

[2] M. Wessel, J. Vargovich, M. A. Gerosa, and C. Treude, "GitHub Actions: The impact on the pull request process," Empir. Softw. Eng., arXiv:2206.14118 [cs.SE], Jul. 2023.

[3] M. Wessel, T. Mens, A. Decan, and P. Rostami Mazrae, "The GitHub development workflow automation ecosystems," arXiv:2305.04772 [cs.SE], May 2023.

[4] A. Chauhan and B. Nayyer, "Optimizing cloud deployment with Boto3: A CI/CD framework for applications," in Proc. 3rd Int. Conf. Networks, Multimedia and Information Technology (NMITCON), 2025, doi: 10.1109/NMITCON65824.2025.11188265.

[5] T. Zhang, S. Pan, Z. Zhang, Z. Xing, and X. Sun, "Deployability-centric infrastructure-as-code generation: Fail, learn, refine, and succeed through LLM-empowered DevOps simulation," arXiv:2506.05623 [cs.SE], v3, Jan. 2026.

[6] D. Mehta, K. Rawool, S. Gujar, and B. Xu, "Automated DevOps pipeline generation for code repositories using large language models," arXiv:2312.13225 [cs.SE], Dec. 2023.

[7] Z. Yang, H. Guan, V. Nicolet, B. Paulsen, J. Dodds, D. Kroening, and A. Chen, "Automated cloud infrastructure-as-code reconciliation with AI agents," arXiv:2510.20211 [cs.SE], Oct. 2025.

[8] A. F. Khan et al., "LADs: Leveraging LLMs for AI-driven DevOps," arXiv:2502.20825 [cs.LG], Feb. 2025.

[9] M. Baqar, S. Naqvi, and R. Khanda, "AI-augmented CI/CD pipelines: From code commit to production with autonomous decisions," in Proc. 3rd Int. Conf. Foundation and Large Language Models (FLLM), 2025, doi: 10.1109/FLLM67465.2025.11391007.

[10] C. Pahl, Ö. C. Sezen, and F. Hofer, "Artificial intelligence for infrastructure-as-code—A systematic literature review," Electronics, vol. 15, no. 4, art. 755, 2026, doi: 10.3390/electronics15040755.

[11] Ł. Chomątek, J. Papuga, P. Nowak, and A. Poniszewska-Marańda, "Decoding CI/CD practices in open-source projects with LLM insights," in Proc. 33rd ACM Int. Conf. Foundations of Software Engineering (FSE Companion '25), Trondheim, Norway, Jun. 2025, doi: 10.1145/3696630.3728699.

[12] F. Calefato, F. Lanubile, and L. Quaranta, "A preliminary investigation of MLOps practices in GitHub," arXiv:2209.11453 [cs.SE], Sep. 2022.

[13] S. Garg, P. Pundir, G. Rathee, P. K. Gupta, S. Garg, and S. Ahlawat, "On continuous integration / continuous delivery for automated deployment of machine learning models using MLOps," arXiv:2202.03541 [cs.SE], Feb. 2022.

[14] U. Cihan, V. Haratian, A. İçöz, M. K. Gül, Ö. Devran, E. F. Bayendur, B. M. Uçar, and E. Tüzün, "Automated code review in practice," in Proc. IEEE/ACM 47th Int. Conf. Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2025, doi: 10.1109/ICSE-SEIP66354.2025.00043.

[15] A. Cepuc, R. Botez, I.-A. Ivanciu, and V. Dobrota, "Implementation of a continuous integration and deployment pipeline for containerized applications in Amazon Web Services using Jenkins, Ansible and Kubernetes," in Proc. 19th RoEduNet Conf.: Networking in Education and Research, 2020, doi: 10.1109/ROEDUNET51892.2020.9324857.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, November 2026
Impact Factor: 8.2

[16] P. Naayini, "Building AI-driven cloud-native applications with Kubernetes and containerization," Int. J. Sci. Adv., vol. 6, no. 2, Mar.–Apr. 2025, doi: 10.51542/ijscia.v6i2.15.
[17] Y. Zhang, H. Li, Y. Zeng, and Z. Wu, "Predictive auto scaling and cost optimization using machine learning in AWS cloud environments," in Proc. ICDIS 2025, Singapore, Sep. 2025, doi: 10.1145/3772326.3774726.
[18] D. Stojanović, L. Vidaković, B. Pavković, and N. Četić, "Comparison of AWS architectures for scalable and cost-efficient retrieval-augmented generation," in Proc. IEEE Zooming Innovation in Consumer Technologies Conf. (ZINC), 2025, doi: 10.1109/ZINC65316.2025.11100283.
[19] R. Karanam, "Multi-cloud IaC template versioning and rollback strategies: An empirical study with Terraform and GitOps," World J. Adv. Res. Rev., vol. 22, no. 2, pp. 2354–2363, 2024, doi: 10.30574/wjarr.2024.22.2.1357.
[20] R. Nagvekar, "Agentic AI-driven CI/CD pipelines for autonomous software delivery," in Proc. 5th IEEE Int. Conf. ICT in Business Industry & Government (ICTBIG), 2025, doi: 10.1109/ICTBIG68706.2025.11323919.
[21] M. E. Barnes, T. A. Ghaleb, and S. Hassan, "From assistance to agency: Rethinking autonomy and control in CI/CD pipelines," in Proc. 3rd ACM Int. Conf. AI-Powered Software (AIware '26), Montreal, QC, Canada, Jul. 2026, doi: 10.1145/3805760.3814897.
[22] B. A. Basha, "AI/ML-driven DevOps automation: Transforming software delivery through intelligent automation," Amer. J. Eng. Technol., vol. 8, no. 8, Aug. 2026, doi: 10.37547/tajet/Volume08Issue08-04.
[23] J. Cui, "LLM-augmented DevOps: Autonomous CI/CD pipeline optimization via reinforcement learning," Solbridge Int. School of Business, Woosong Univ., Daejeon, South Korea, unpublished manuscript.
[24] E. Kazlauskaitė, "An AI-driven DevOps framework for scalable multi-cloud software deployment and lifecycle maintenance: An analytical study," Eur. J. Emerg. Data Sci. Mach. Learn., vol. 3, no. 1, pp. 47–51, May 2026.
[25] R. Namburi, "Intelligent DevOps: Leveraging machine learning for automated CI/CD, monitoring, and cloud resource optimization in Java-based systems," in Proc. Int. Conf. Computing Theory and Wireless Communications (ICCTWC), Ichalkaranji, India, Apr. 2026, doi: 10.1109/ICCTWC68241.2026.11583747.
[26] L. G. Felipe, "Agentic AI software deployment: Enhancing software deployment with agentic AI for autonomous and scalable operations," M.S. thesis, NOVA Inf. Manage. School, Univ. Nova de Lisboa, Lisbon, Portugal, Apr. 2026.
[27] X. Li, S. Lu, F. Sarro, C. Le Goues, and H. Ye, "HerAgent: Rethinking the automated environment deployment via hierarchical test pyramid," arXiv:2602.07871 [cs.SE], Feb. 2026.
[28] S. Sudarsan, M. Agrawal, and A. Banerjee, "AI-assisted detection of malicious changes in infrastructure-as-code for secure DevOps," in Proc. 5th IEEE Int. Conf. AI in Cybersecurity (ICAIC), Houston, TX, USA, Feb. 2026, doi: 10.1109/ICAIC67076.2026.11395715.
[29] A. Mittal and V. Venkatesan, "Practical integration of large language models into enterprise CI/CD pipelines for security policy validation: An industry-focused evaluation," in Proc. IEEE Int. Conf. Service-Oriented System Engineering (SOSE), 2025.
[30] M. Esposito, M. Robredo, A. Bakhtin, D. Taibi, and V. Lenarduzzi, "Generative AI as an infrastructure copilot: Automating infrastructure-as-code across the DevSecOps lifecycle," Autom. Softw. Eng., vol. 33, art. 58, 2026, doi: 10.1007/s10515-026-00600-5.