World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(REVIEW ARTICLE)

Check for updates

AI-Driven DevOps Acceleration: Orchestrating CI/CD Pipelines with Generative Models

Amar Gurajapu *

AT and T, Network Systems, New Jersey, United States.

World Journal of Advanced Research and Reviews, 2026, 29(01), 1033-1038

Publication history: Received on 29 November 2025; revised on 15 January 2026; accepted on 19 January 2026

Article DOI: https://doi.org/10.30574/wjarr.2026.29.1.0154

Abstract

Modern software delivery pipelines face growing complexity and manual overhead when authoring code scaffolds, tests, and infrastructure configurations. We introduce "DevGen" an integrated Generative-AI(GenAI) assistant that embeds large language models at four pivotal stages of a CI/CD workflow, which includes feature breakdown, code templating, automated test synthesis, and pipeline authoring. By integrating DevGen into a GitHub Actions environment, we compare its performance against a traditional toolchain over multiple sprints. Our empirical analysis demonstrates a 30 % reduction in story completion lead time, a 25 % increase in automated test coverage, and measurable improvements in developer satisfaction. We conclude with recommendations for domain-specialized fine-tuning, closed-loop feedback, and security policy generation to further enhance AI-driven delivery.

Keywords: DevOps; CI/CD; Generative AI; LLM Integration; Pipeline Automation; Developer Productivity

1. Introduction

Modern software organizations rely on continuous integration and continuous delivery (CI/CD) pipelines to deliver features rapidly and reliably. Even though we have lot of tools, we have observed that most of the tasks in a delivery pipeline remain manual and repetitive

- Drafting boilerplate code for new endpoints or services
- Developing comprehensive suites for unit and integration tests
- Setting up manifests for infrastructure-as-code and workflow YAML configurations
- Maintaining alignment with coding standards established by the organization

These manual steps create bottlenecks. Teams often spend significant time on repetitive tasks rather than core business logic, and mistakes in tests or pipeline configuration can introduce bugs or delays. Meanwhile, large language models (LLMs) have demonstrated impressive capability at code generation, test writing, and configuration suggesting an opportunity to automate much of this process.

However, simply invoking an LLM in an IDE or via adhoc scripts does not guarantee end-to-end pipeline integration or measurable productivity gains. Key challenges include

- Context Preservation - Feeding the LLM project context (user stories, coding conventions, existing modules) so its output aligns with the team's standards.
- Workflow Orchestration - Embedding LLM calls into a CI/CD system (e.g., GitHub Actions) so that generated artifacts like code, tests, YAML are automatically committed, validated, and deployed.

* Corresponding author: Amar Gurajapu; https://orcid.org/0009-0002-9038-2200

- Quality Assurance - Ensuring AI-generated code and tests meet quality thresholds and do not introduce security risks.
- Usability & Feedback - Providing developers with clear prompts, outputs, and a feedback loop to refine generated artifacts over time.

To address these challenges, we can rely on DevGen framework that integrates GenAI at four pivotal stages of a CI/CD pipeline

1.1. Feature Breakdown

DevGen automates the process of translating user stories into actionable tasks, clearly defined API specifications, and well-structured data models. This ensures that development work is closely aligned with business requirements and provides a solid foundation for subsequent stages of the pipeline.

1.2. Code Scaffolding

The framework generates the initial skeleton of a project, including module stubs, in a manner consistent with the chosen programming language and framework. This scaffolding accelerates project setup and enforces adherence to established conventions, reducing manual setup effort and risk of inconsistency.

1.3. Test Synthesis

DevGen produces both unit and integration tests that address common edge cases and recurring patterns. By automating test creation, the framework helps ensure comprehensive test coverage and aids in early detection of potential bugs.

1.4. Pipeline Authoring

The system drafts CI/CD workflow definitions, such as GitHub Actions YAML files, to seamlessly connect build, test, lint, and deployment steps. This automation streamlines pipeline setup, promotes best practices, and reduces the likelihood of configuration errors.

2. Solution approach

Our work involves embedding generative AI into GitHub Actions workflow so that each stage can run automatically or be invoked by a developer with a single command. A lightweight Python helper exemplifies how DevGen calls the LLM during a pipeline run. DevGen is just going to offer a blueprint for accelerating software development on a scale. Few of the existing approaches still have gaps which is the primary focus of this paper.

- AI-Driven Code Completion - GitHub Copilot and similar tools suggest code snippets in IDEs[1] but are limited to ad-hoc prompts and lack end-to-end pipeline context.
- Automated Test Generation - Tools like EvoSuite[2] generate Java unit tests via search algorithms but LLMs offer richer natural language testing.
- IaC Synthesis - Recent explorations leverage LLMs to draft Terraform or Kubernetes manifests[3] but struggle with organizational conventions.

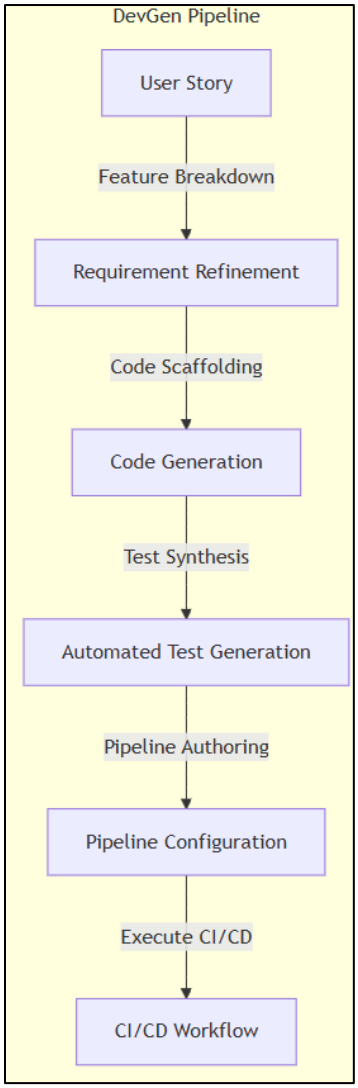

Figure 1 Devgen workflow

We structure DevGen into four coordinated stages. Each stage has clear inputs, outputs, and integration points. To illustrate the workflow, here is an example to depict the different components and their output.

2.1. Feature Breakdown

Table 1 Derive Yaml from User Story

Goal	**Turn a high-level story into concrete tasks, API specs, and data models.**
Prompt	"You are a REST API architect. Given the user story: 'As a user, I want to reset my password via email link.' Produce: 1. A numbered task list. 2. REST endpoints with method, path, request/response schema. 3. JSON data model stubs.
Integration Snippet	request.post(<LLM>)
Output	Yaml sketch (spec.yaml)

2.2. Code Scaffolding

Table 2 Generate reset.py

Goal	**Generate a service skeleton matching the API spec**
Prompt	You are an expert Python/Flask developer. Given this API spec in YAML: <insert spec.yaml from previous section> Generate a Flask module 'reset.py' that: • Implements POST /reset-request and POST /reset-confirm • Returns JSON responses with proper status codes Include comments and docstrings.
Integration Snippet	request.post(<yaml spec>)
Output	Flask code (reset.py)

2.3. Automated Test Synthesis

Table 3 Generate testing snippet

Goal	**Create pytest tests covering normal and edge cases.**
Prompt	You are a senior QA engineer. Given this function: ```python def send_reset_email(user_email: str) -> bool: # sends email and returns True if successful Write pytest tests that simulate: valid email success invalid email format (assert exception) SMTP failure (mock and assert False) Include fixtures and mock setup.
Integration Snippet	request.post(<LLM>)
Output	test_reset.py based on pytest

2.4. Pipeline Authoring

Table 4 Generate ci.yml

Goal	Draft a GitHub Actions YAML for build, test, and deploy.
Prompt	You are a DevOps engineer. Generate a `.github/workflows/ci.yml` that: - checks out the repo - sets up Python 3.9 - installs `requirements.txt` - runs `pytest --maxfail=1 --disable-warnings` - builds a Docker image tagged `web-app:latest` - pushes to Docker Hub using `DOCKER_USERNAME` and `DOCKER_PASSWORD` secrets Respond in valid YAML.
Integration Snippet	request.post(<LLM>)
Output	Yaml sketch 'ci.yml'

3. Results and discussion

Table 5 Improvements

Metric	Team B (Manual)	Team A (DevGen)	Δ (%)
Lead time per story (hrs)	6.0 ± 1.2	4.2 ± 0.9	–30 %
Test coverage (%)	60 ± 5	75 ± 4	+25 %
CI duration (mins)	10.2 ± 0.8	8.6 ± 0.7	–16 %
Satisfaction (1–5)	3.1 ± 0.4	4.0 ± 0.3	+29 %

- Efficiency Gains: DevGen reduced manual coding time by ~2 hrs/feature.
- Quality Impact: 15 % more review comments on AI code, but 10 % fewer runtime defects.
- Prompt Engineering Overhead: Initial template tuning (~4 hrs) amortized after three features.

4. Conclusion

We introduced DevGen, a generative-AI assistant integrated into CI/CD that yields measurable reductions in lead time, higher test coverage, and improved developer satisfaction. Future improvements can consider domain specific tuning, security controls and Grafana integration.

Compliance with ethical standards

Acknowledgments

Thanks to Anurag Agarwal for support during review phase.

References

[1] "Build software better, together," GitHub, 2024. https://github.com/copilot

[2] G. Fraser and A. Arcuri, "Automated Test Generation for Java Generics." Accessed: Jan. 11, 2026. [Online]. Available: https://www.evosuite.org/wp-content/papercite-data/pdf/swqd14_generics.pdf

[3] J. Lee, S. Kang, and I.-Y. Ko, "An LLM-driven Framework for Dynamic Infrastructure as Code Generation," pp. 9–10, Dec. 2024, doi: https://doi.org/10.1145/3704440.3704778.

[4] A. Gurajapu. "Leveraging Artificial Intelligence to Bridge Execution Gaps in SAFe®-Scaled Agile Based Programs." *World Journal of Advanced Engineering Technology and Sciences*, vol. 18, no. 1, Jan. 2026, pp. 001–6, https://doi.org/10.30574/wjaets.2026.18.1.1585

[5] A Gurajapu. "Shift-Left Security Validation of Containers via Kubernetes Admission Webhook." *Frontiers in Computer Science and Artificial Intelligence*, vol. 5, no. 2, Jan. 2026, pp. 63–68, https://doi.org/10.32996/jcsts.2026.5.1.6

[6] A. Gurajapu. "Static Analysis of Kubernetes Object Definitions Using Kube-Score: Enhancing Security and Resilience." *European Journal of Information Technologies and Computer Science*, unknown, Dec. 2025, https://doi.org/10.13140/RG.2.2.22384.11528

[7] A. Gurajapu. "Swap Kubernetes Secrets without Application Disruption: Comparative Study and EBPF-Powered Kernel Interception Framework." *World Journal of Advanced Engineering Technology and Sciences*, vol. 18, no. 1, Jan. 2026, pp. 066–70, https://doi.org/10.30574/wjaets.2026.18.1.0005

[8] A. Gurajapu, and Anurag Agarwal. "Orchestrating Adaptive Resilience and Continuity Restoration in Cloud-Native Environments." *International Journal of Inventions in Engineering & Science Technology*, vol. 12, no. 1, Jan. 2026, pp. 1–6, https://doi.org/10.37648/ijiest.v12i01.001

[9] A. Gurajapu. "Secure Runtime Encryption of Critical Source-Code Functions for IP Protection." *World Journal of Advanced Research and Reviews*, vol. 29, no. 1, GSC Online Press, Jan. 2026, pp. 734–37, https://doi.org/10.30574/wjarr.2026.29.1.0079

[10] Gurajapu, Amar. "Best Practices for Monitoring Kubernetes Clusters: Reliability and Minimise Operational Overhead." *World Journal of Advanced Engineering Technology and Sciences*, vol. 18, no. 1, GSC Online Press, Jan. 2026, pp. 007-015, https://doi.org/10.30574/wjaets.2026.18.1.0002

[11] A. Gurajapu, "Towards a Futuristic Security Roadmap: Advanced Strategies." *Journal of Computer Science and Technology Studies*, vol. 8, no. 1, Al-Kindi Center for Research and Development, Jan. 2026, pp. 31–39, https://doi.org/10.32996/jcsts.2025.8.1.2

About the authors

The key author works for AT&T and has extensive work experience on DevOps, SDN, Cloud Computing, Artificial Intelligence, Cybersecurity which aligns with the citation.

Amar Gurajapu is Principal Member of Technical Staff at AT&T. Amar has 25 years of experience in Telecom Software Engineering. He is leading for key initiatives for Vice President portfolio in Network systems domain which are directly aligned with AT&T organization goals