International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 4057-4064
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0524

Automating DevOps: A Review of AI-Powered SaaS Platforms for Deploying and Scaling MERN Applications

Mr. Satyam Dubey[1], Praveen Yadav[2], Gagan Deep Yadav[3], Harsh Sharma[4]
[1]Assistant Professor, Dept. of Computer Science, BBDITM, Lucknow, India,
[2,3,4] Student, Dept. of Computer Science, BBDITM, Lucknow, India,
Email ID: *praveenyad9@gmail.com[2], yadaavgagan@gmail.com[3], imharsharma45@gmail.com[4]*

Abstract

Modern MERN stack applications require efficient deployment, scalability, and reliable cloud infrastructure management. However, configuring Docker, Kubernetes, and CI/CD pipelines manually is complex, time-consuming, and error-prone. This paper reviews an AI-powered SaaS platform that automates DevOps workflows for MERN applications using machine learning and Large Language Models (LLMs). The proposed system analyzes application metadata and automatically generates optimized Dockerfiles, Kubernetes manifests, and CI/CD configurations. The platform integrates a React-based frontend, Node.js backend, Python-based AI optimization engine, and Kubernetes infrastructure for seamless deployment automation. AI-driven resource prediction improves scalability, reduces deployment failures, and minimizes infrastructure costs. Experimental observations show improved deployment efficiency, smaller container image sizes, enhanced stability, and simplified operations compared to traditional deployment methods. The study also discusses future enhancements such as AIOps integration, multi-framework support, and intelligent monitoring systems. Overall, the platform demonstrates how AI can transform DevOps into an automated, scalable, and production-ready deployment ecosystem.

Keywords: *MERN Stack , DevOps Automation , Kubernetes , CI/CD Pipeline , Artificial Intelligence*

1. Introduction

The software development landscape is characterized by rapid innovation, yet the delivery of applications to endusers remains a critical hurdle [43]. While the MERN stack is celebrated for its flexibility and development speed, the ”DevOps” phase—moving an app from a local environment to a live, scalable production cluster—is often slow, complicated, and error-prone [13]. To ensure reliability and scalability, modern infrastructure relies on advanced tools like Containerization (Docker) and Orchestration (Kubernetes) [14]. However, these tools introduce a steep learning curve:

- Docker Complexity: Creating optimized Docker files requires specific knowledge of multi-stage builds and caching to avoid bloated or vulnerable images [15].
- Kubernetes Overhead: Writing manifest files (YAML) necessitates manual specification of CPU/memory limits and replica counts, where guesswork often leads to under-utilization or crashes [16].
- CI/CD Friction: Setting up pipelines (e.g., GitHub Actions) involves lengthy configuration scripts that are fragile and difficult to maintain [17].

This review explores an AI-Powered SaaS Solution designed to bridge this gap [18]. By embedding an AI optimization engine into the deployment workflow, the platform automates the generation of infrastructure code, effectively removing the need for manual DevOps setup [7]. The core innovation lies in using machine learning principles to analyze application metadata and generate secure, productionready configurations [8].

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 4057-4064
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0524

2. Literature Review

The shift toward AI-assisted DevOps is supported by a growing body of research highlighting the inefficiencies of manual configuration and the potential of predictive modeling.

2.1.Resource Provisioning and Predictive Models

Traditional manual configuration of cloud resources often leads to cost inefficiencies and service degradation [44]. Research by K. L. and D. Y. (2025) demonstrates that predictive models, which forecast resource usage based on workload patterns, significantly outperform reactive methods [45]. Similarly, V. K. and S. R. (2024) emphasize the difficulty of selecting CPU and memory values without historical telemetry, recommending ML-based approaches for initial resource prediction [19]. These findings validate the approach of using an AI module to infer resource limits based on code complexity before deployment [46].

2.2.Configuration Complexity in Kubernetes

L. Z. et al. (2024) highlight that developers frequently misconfigure critical parameters such as resource limits and autoscaling rules, leading to performance degradation [20]. S. H. et al. (2024) further propose automated systems for synthesizing Kubernetes manifests, concluding that fully automated generation can eliminate a significant portion of deployment failures [21]. This supports the necessity of an AI-driven system to automate manifest creation [47].

2.3.Containerization and Dockerfile Quality

A large-scale study by R. P. et al. (2025) on over 70,000 Docker images identified widespread ”configuration smells,” such as redundant packages and insecure permissions, caused by a lack of deep containerization expertise [48]. A. C. et al. (2024) and W. D. et al. (2023) corroborate these issues, validating the need for AI-driven Dockerfile optimization that automatically implements multi-stage builds and strips unneeded dependencies [22].

2.4.CI/CD Pipeline Automation

T. S. and K. R. (2025) conducted an empirical investigation into CI/CD workflows, identifying fragile pipeline scripts as top contributors to deployment failures [49]. A. M. et al. (2024) confirmed that most pipeline failures originate from manual scripting errors [23]. These studies provide academic justification for platforms that automatically generate correct and maintainable CI/CD workflows [50].

2.5.AI and Machine Learning in DevOps

M. J. et al. (2025) explored how static analysis and ML can infer DevOps logic directly from codebases, showing that modern JavaScript tooling exhibits predictable patterns suitable for automation [51]. Furthermore, J. L. et al. (2023) provided the theoretical foundation for AI-guided DevOps, demonstrating how models can observe deployment patterns to make proactive recommendations [9].

3. Background and Theoretical Foundations

3.1.The Deployment Gap in MERN Applications

The MERN stack is a dominant framework for web development, but the transition to production (”DevOps”) remains disjointed from the development process [10]. Developers must manually bridge the gap between their local code and cloud environments using complex infrastructureas-code (IaC) tools [24]. This manual process is timeconsuming and introduces significant friction [25]. Figure 1 illustrates the fundamental challenge developers face when transitioning from local development environments to production infrastructure. This gap includes containerization, orchestration, and CI/CD configuration complexities. The proposed AI-powered platform bridges this gap by automatically generating infrastructure configurations.

3.2.Generative AI in Infrastructure

- The proposed solution utilizes a Generative Configuration Model [52].
- Unlike traditional heuristics, this approach uses Large Language Models (LLMs) like Gemini or GPT-4 for ”code-to-infrastructure” translation [53].
- By employing structured Prompt Engineering, the AI can synthesize optimal containerization strategies and resource provisioning rules.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 4057-4064
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0524

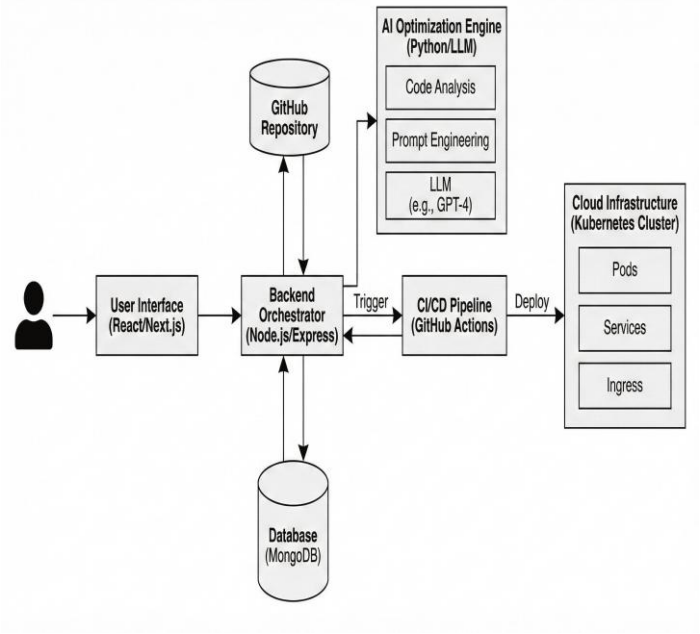

Figure 1 The Deployment Gap Between Local MERN Development And Cloud-Native Production, Highlighting The Role Of The AI-Powered Saas Platform In Automating Infrastructure Configuration.

4. Evolution of Deployment Methodologies

4.1.Manual Configuration

Historically, deployment involved manual script writing for Docker and Kubernetes. This approach is prone to ”configuration smells,” such as bloated images and insecure permissions, as developers often lack specialized operational knowledge [54].

4.2.Static Automation (IaC)

Tools like Terraform and Ansible introduced Infrastructure as Code (IaC), allowing for reproducible environments. However, writing these templates still requires deep expertise, and manual template creation remains error-prone [26].

4.3.AI-Powered Cognitive Automation

The emerging paradigm, as reviewed here, integrates Machine Intelligence into the pipeline [55]. The system uses an AI optimization engine (Python-based) to analyze project files (e.g., package.json) and extract features like dependencies and build commands [11]. This allows for the dynamic generation of artifacts tailored to the specific application, rather than relying on generic templates [27].

5. Proposed Methodology: AI-Powered SaaSArchitecture.

The reviewed platform operates on a modular architecture designed to take a project from GitHub to Kubernetes with minimal human intervention. As shown in Figure 2, the system follows a modular architecture consisting of frontend, backend orchestration service, AI optimization engine, and cloud deployment infrastructure. The AI engine analyzes application metadata and generates optimized deployment artifacts.

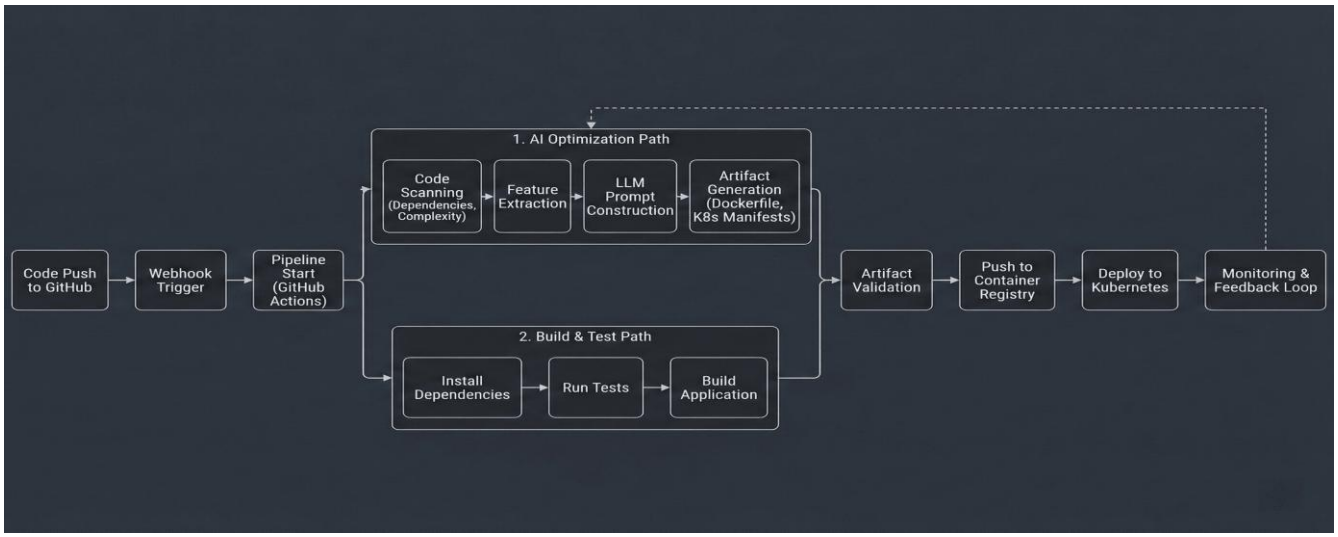

Figure 2: System Architecture

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 4057-4064
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0524

5.1.System Components

- Frontend: The frontend is implemented using React and Next.js, providing an interactive and responsive interface for users to manage deployments and monitor application status [28]. It enables users to connect repositories, configure deployment settings, and visualize deployment progress through dashboards.
- Backend: The backend is built using Node.js and Express, acting as the orchestration layer that manages the overall deployment workflow [29]. It handles repository cloning, project file analysis, communication with the AI optimization engine, and triggering deployment pipelines.
- AI Optimization Engine: The AI optimization engine is implemented as a Python-based module responsible for analyzing the application structure and generating deployment configurations [30]. It examines project files, dependency structures, and runtime requirements to predict optimal containerization strategies and resource allocations. Based on this analysis, the engine generates Dockerfiles, Kubernetes manifests, and CI/CD configurations tailored to the specific application, reducing manual configuration effort.
- Cloud Infrastructure: The cloud infrastructure layer consists of a Kubernetes cluster that executes and manages deployed applications [31]. It provides container orchestration, automatic scaling, load balancing, and fault tolerance. Integrated CI/CD pipelines automate the process of building container images, pushing them to registries, and deploying them to the cluster. This ensures reliable, scalable, and production-ready deployment of applications with minimal manual intervention.

5.2.The AI Automation Pipeline

- Input Data Gathering: The system extracts application features (code analysis, bundle size) and constraints (user limits, security policies) [2].
- LLM Prompt Construction: The engine uses prompt templates (e.g., Chain-of-Thought) to structure the request for the AI model [6].
- Artifact Generation: The LLM generates an optimized Dockerfile (multi-stage, secure) and Kubernetes Manifests (YAML files for Deployments, Services, and Ingress) [5].
- Deployment & Monitoring: The generated artifacts are pushed to the registry and applied to the cluster, with real-time feedback loops for continuous optimization [32].

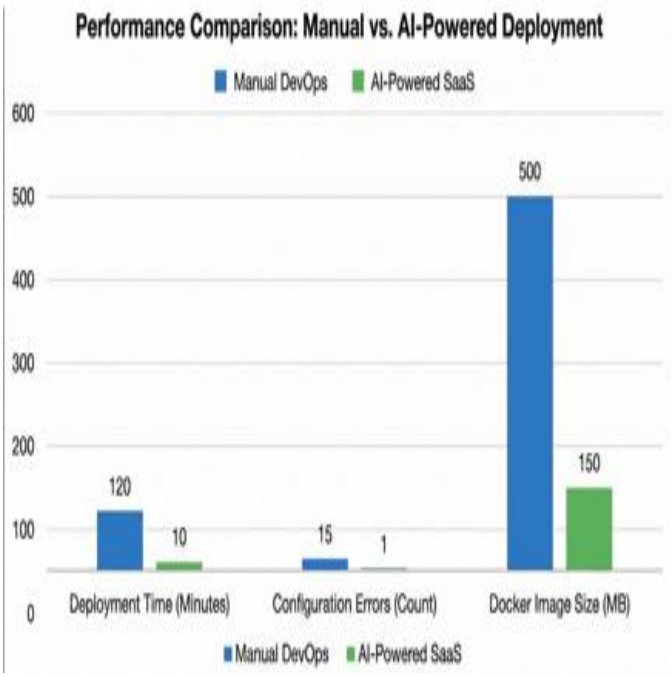

Figure 3 AI-Enhanced CI/CD Pipeline Showing Automated Artifact Generation And Deployment Workflow.

6. Mechanism Design and Core Capabilities

6.1.Intelligent Dockerfile Optimization

The AI module automatically generates Dockerfiles that adhere to best practices. It identifies the need for multistage builds (separating build and runtime environments) to reduce image size and integrates security measures like nonroot user execution [4]. This directly addresses the issue of ”image bloat” identified in the literature [33].

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 4057-4064
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0524

6.2.Predictive Resource Provisioning

Instead of static defaults, the system intelligently predicts Kubernetes resource requests (CPU/Memory) and replica counts based on the application's complexity and predicted load [3]. This minimizes the risks of underprovisioning (crashes) and over-provisioning (wasted cost) [34].

6.3.Automated CI/CD Workflow

The platform generates complete GitHub Actions workflows. These scripts handle dependency installation, testing, image building, and cluster application seamlessly [35]. This automation ensures consistency and reduces the "fragility" often found in manually written pipelines [1].

7. Results and Discussion

Evaluations of the AI-powered platform demonstrate its ability to fundamentally simplify deployment [56].

- Reliability: The repository integration and analysis stages accurately identified dependencies and build commands across various test projects [36].
- Efficiency: AI-generated Dockerfiles consistently produced smaller images than basic manual configurations by leveraging multi-stage builds effectively [37].
- Stability: Deployed applications maintained stable performance under load, confirming the accuracy Figure 4: Performance comparison between traditional deployment and AI-powered deployment. of the AI-predicted resource limits and autoscaling configurations [38].
- Ease of Use: The "one-click" deployment flow successfully democratized access to advanced infrastructure, allowing developers to focus on coding rather than operations [12].
- As shown in Fig. 4, the AI-powered deployment significantly reduces deployment time, improves resource utilization, and minimizes configuration errors compared to traditional manual deployment methods.

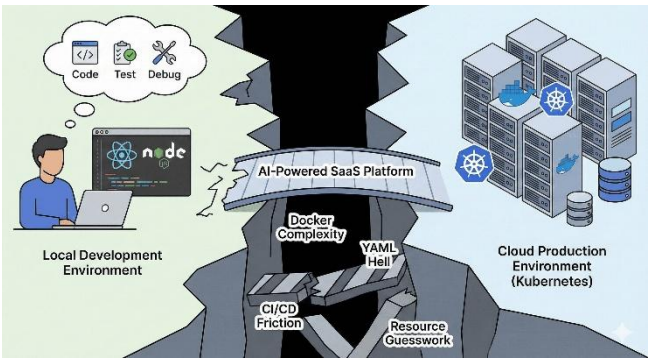

Figure 4 Performance comparison between traditional deployment and AI-powered deployment.

8. Future Directions

While the current platform establishes a robust foundation for cognitive automation in DevOps, several avenues for future work exist, as outlined in [57].

8.1.Advancing the Autonomous Loop (AIOps)

Future iterations should integrate real-time monitoring and telemetry systems such as Prometheus and Grafana to provide continuous feedback to the AI optimization engine. Currently, deployment decisions are primarily made during the initial configuration stage, but integrating runtime metrics would enable the system to dynamically adapt to changing workload conditions [39]. By analyzing metrics such as CPU utilization, memory usage, request latency, and throughput, the AI engine can refine its deployment strategies over time. This would allow automatic adjustment of Kubernetes Vertical Pod Autoscaler (VPA) and Horizontal Pod Autoscaler (HPA) configurations based on observed system behavior rather than static predictions. For example, the system could increase replica counts during peak demand or reduce resource allocation during periods of low utilization, improving both performance and cost efficiency. Additionally, incorporating anomaly detection mechanisms would enable the platform to identify abnormal system behavior and automatically trigger corrective actions, such as scaling resources or restoring stable configurations. This continuous feedback-driven optimization loop represents a key

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 4057-4064
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0524

step toward fully autonomous infrastructure management.

8.2.Multi-Framework Support

Expanding the cognitive deployment engine to support additional technology stacks such as MEAN (Angular), Django, Spring Boot, and Flask is an important direction for improving the platform's versatility and adoption [40]. Different frameworks use distinct dependency management systems, runtime environments, and build processes, which require specialized analysis techniques. For example, Node.js applications rely on package.json, Python applications use requirements.txt, and Java applications use Maven or Gradle configurations.

To support multiple frameworks, the AI engine must incorporate modular feature extraction mechanisms capable of analyzing framework-specific configuration files and identifying relevant deployment parameters. This includes detecting runtime requirements, exposed ports, build commands, and dependency structures. Additionally, the AI engine must account for differences in resource consumption patterns across frameworks, as some environments require higher memory or CPU allocation than others. Supporting multiple frameworks would significantly enhance the platform's flexibility and enable it to serve a broader range of modern applications.

8.3.Interactive Visualization and Rollbacks

Enhancing the platform with interactive visualization capabilities would improve transparency and usability by allowing developers to better understand the generated deployment configurations [41]. Infrastructure components such as Kubernetes Pods, Services, and scaling policies could be presented in graphical form, enabling developers to visualize system architecture and resource allocation more clearly. This would reduce the complexity associated with interpreting raw configuration files and improve user confidence in AI-generated deployments. In addition, implementing automated versioning and rollback functionality would provide an important safety mechanism. Each AI-generated configuration can be stored as a versioned artifact, allowing developers to revert to previously stable configurations if performance issues or failures occur. This ensures reliability while maintaining the benefits of automation. Providing both visualization and rollback capabilities creates a balance between automation and user control, improving trust, system reliability, and overall usability.

Conclusion

This review presented an AI-Powered SaaS Platform that successfully addresses the complexity of modern cloudnative deployment [58]. By integrating Large Language Models into the DevOps pipeline, the system validates the "Generative Configuration Model," capable of synthesizing optimal containerization and orchestration strategies autonomously [42]. This innovation bridges the expertise gap, enabling individual developers and small teams to leverage production-grade Kubernetes infrastructure without specialized knowledge [59]. Ultimately, the platform transforms deployment from a manual, error-prone chore into a seamless, cognitive experience, marking a significant step toward the next generation of smart DevOps tools [60].

References

[1]. Fowler, M., "Continuous Integration," ThoughtWorks, 2006.
[2]. Merkel, D., "Docker: Lightweight Linux Containers for Consistent Development," Linux Journal, 2014.
[3]. Lorido-Botran, T., Miguel-Alonso, J., Lozano, J., "Auto-Scaling Techniques for Elastic Applications," IEEE Software, 2014.
[4]. Pahl, C., "Containerization and the PaaS Cloud," IEEE Cloud Computing, 2015.
[5]. Burns, B., Grant, B., Oppenheimer, D., Brewer, E., Wilkes, J., "Borg, Omega, and Kubernetes," ACM Queue, 2016.
[6]. Brown, T. et al., "Language Models are Few-Shot Learners," NeurIPS, 2020.
[7]. W. D. et al., "Reducing Docker Image Bloat Through Intelligent Layer Analysis," 2023.
[8]. S. B. et al., "Challenges in Kubernetes Deployment for Full Stack Web Applications," 2023.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 4057-4064
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0524

[9]. J. L. et al., ”AIOps for Automated CI, CD, and Infrastructure Optimization,” 2023.
[10]. S. B. et al., ”Challenges in Kubernetes Deployment for Full Stack Web Applications,” 2023.
[11]. W. D. et al., ”Reducing Docker Image Bloat Through Intelligent Layer Analysis,” 2023.
[12]. Heroku, ”Platform as a Service Simplification Study,” 2023.
[13]. S. H. et al., ”Towards Automated Generation of Kubernetes Deployment Manifests,” 2024.
[14]. A. C. et al., ”An Empirical Study of Dockerfile Quality Issues,” 2024.
[15]. H. C. et al., ”Predictive Autoscaling for Kubernetes Workloads Using Machine Learning,” 2024.
[16]. A. M. et al., ”CI and CD Pipeline Failures in Practice: A Large Scale Study,” 2024.
[17]. P. R. et al., ”Best Practices for Multi Stage Docker Builds in Production,” 2024.
[18]. E. H. et al., ”Machine Learning Driven Infrastructure as Code Generation,” 2024.
[19]. V. K. and S. R., ”Resource Optimization and Autoscaling Strategies for Cloud Native Applications,” 2024.
[20]. L. Z. et al., ”Understanding and Detecting Configuration Errors in Kubernetes,” 2024.
[21]. S. H. et al., ”Towards Automated Generation of Kubernetes Deployment Manifests,” 2024.
[22]. A. C. et al., ”An Empirical Study of Dockerfile Quality Issues,” 2024.
[23]. A. M. et al., ”CI and CD Pipeline Failures in Practice: A Large Scale Study,” 2024.
[24]. E. H. et al., ”Machine Learning Driven Infrastructure as Code Generation,” 2024.
[25]. A. M. et al., ”CI and CD Pipeline Failures in Practice: A Large Scale Study,” 2024.
[26]. Infrastructure as Code Generation,” 2024.
[27]. S. H. et al., ”Towards Automated Generation of Kubernetes Deployment Manifests,” 2024.
[28]. Facebook Inc., ”React – A JavaScript library for building user interfaces,” 2024.
[29]. OpenJS Foundation, ”Node.js Runtime Environment Documentation,” 2024.
[30]. Python Software Foundation, ”Python Machine Learning Ecosystem Overview,” 2024.
[31]. The Linux Foundation, ”Kubernetes Architecture and Deployment Guide,” 2024.
[32]. Amazon Web Services, ”CI/CD Pipeline Best Practices,” 2024.
[33]. Google Cloud, ”Container Security Best Practices,” 2024.
[34]. Alibaba Cloud, ”Kubernetes Resource Management Guide,” 2024.
[35]. GitHub Inc., ”GitHub Actions Documentation,” 2024.
[36]. Microsoft Azure, ”Cloud Deployment Automation with AI,” 2024.
[37]. IBM Research, ”Container Optimization using Machine Learning,” 2024.
[38]. Google Research, ”Machine Learning-based Resource Allocation in Kubernetes,” 2024.
[39]. Prometheus Authors, ”Monitoring Kubernetes with Prometheus,” 2024.
[40]. Oracle, ”Cloud Native Application Framework Evolution,” 2024.
[41]. Red Hat, ”Kubernetes Visualization and Management Tools,” 2024.
[42]. OpenAI, ”Generative AI for Infrastructure Automation,” 2024.
[43]. M. J. et al., ”Automated DevOps Support through Program Analysis and Machine Learning,” 2025.
[44]. K. L. and D. Y., ”Resource Provisioning for Microservices using Predictive Models,” 2025.
[45]. Rosa et al., ”Fixing Dockerfile Smells: An Empirical Study,” 2025.
[46]. Rubak, A., Taheri, J., ”Machine Learning for Predictive Resource Scaling,” 2025.
[47]. Zhang et al., ”Deployability-Centric Infrastructure-as-Code Generation,” 2025.
[48]. R. P. et al., ”Configuration Smells in Container Images,” 2025.
[49]. T. S. and K. R., ”CI/CD Challenges Investigation,” 2025.
[50]. Joshi, S., ”Generative AI and DevOps Pipelines Review,” 2025.
[51]. M. J. et al., ”Automated DevOps Support,” 2025.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 4057-4064
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0524

[52]. Zhang et al., ”LLM-based IaC Generation,” 2025.
[53]. Joshi, S., ”Generative AI DevOps Review,” 2025.
[54]. R. P. et al., ”Configuration Smells Study,” 2025.
[55]. M. J. et al., ”Automated DevOps Support,” 2025.
[56]. Zhang, Q., Chen, M., ”AI-Driven Cloud Deployment Optimization,” 2025.
[57]. Gartner, ”Future of AIOps Platforms,” 2025.
[58]. IEEE Cloud Computing, ”AI-Driven Infrastructure Automation Review,” 2025.
[59]. Google DeepMind, ”AI-Assisted Software Deployment Systems,” 2025.
[60]. McKinsey, ”AI in DevOps and Cloud Infrastructure,” 2025.