The American Journal of Engineering and Technology
ISSN 2689-0984 Volume 08 - 2026

AI/ML-Driven DevOps Automation: Transforming Software Delivery Through Intelligent Automation

Basheer Ahmedu Basha

Principal Build and Release Engineer - IEEE

Received: 09 June 2026 | Received Revised Version: 28 July 2026 | Accepted: 04 Aug 2026 | Published: 18 Aug 2026

Volume 08 Issue 08 2026 | Crossref DOI: 10.37547/tajet/Volume08Issue08-04

Abstract

The integration of Artificial Intelligence (AI) and Machine Learning (ML) is transforming DevOps, enabling it to become a predictive, intelligent, and adaptive process across the software delivery lifecycle. This study aims to create an AI/ML based DevOps automation framework for deployment risk prediction, deployment optimization, rollback management, and continuous monitoring in a single CI/CD workflow. The framework's data components are the build logs, deployment history, infrastructure monitoring, configuration repositories, incident records, and user feedback, which are then combined with DevOps data into a machine learning pipeline. The data is then preprocessed, combined, and processed by feature engineering to be split into training, validation, and testing sets. A Random Forest model is employed for deployment risk prediction, while an AI-driven decision engine uses predicted risk and operational conditions to support deployment optimization, resource allocation, automated rollback, and continuous monitoring. Experimental evaluation demonstrates a 50% reduction in average build time, a 60% reduction in deployment failure rate, a 50% reduction in rollback frequency, a 34% improvement in deployment risk prediction accuracy, and a 68% reduction in false positive rate. The findings demonstrate improved software delivery efficiency, reliability, operational stability, and decision-making.

Keywords: Artificial Intelligence, Machine Learning, DevOps Automation, Continuous Monitoring, Intelligent Software Delivery.

Cite This Article: Basha, B. A. (2026). AI/ML-Driven DevOps Automation: Transforming Software Delivery Through Intelligent Automation. The American Journal of Engineering and Technology, 8(08), 31–42. https://doi.org/10.37547/tajet/Volume08Issue08-04

I. Introduction

Software is the catalyst for digital transformation, allowing businesses to build new products and services in a range of industries, including healthcare, finance, manufacturing, education, retail, transportation, and telecommunications. The complexity of modern software systems has grown as their use of cloud computing, microservices [1][2], IoT and distributed computing has been scaled up [3][4]. As a result, organizations are expected to provide high-quality software in a timely fashion, with reliability, scalability, security, and continuous availability. This expectation can only be met if the software delivery process is efficient, can accommodate regular software changes, reduces the effects of operations, and, delivers consistent software

performance [5][6]. The adoption of digital technologies is happening rapidly in the world of business, and meeting the demand for faster software delivery without sacrificing quality is a top priority for software engineering teams and organizations.

DevOps has become a popular software engineering practice that stitches software development and IT operations together into a single process to meet these demands [7][8]. DevOps has several key principles and practices, including CI/CD [9], Infrastructure automation, automated testing, continuous monitoring, and collaborative development practices, which allow companies to launch software faster and more reliably than traditional software delivery methods [10][11]. Automated pipelines reduce deployment time, simplify communications between dev and ops teams, and provide software quality by continuous validation and monitoring [12][13]. Even though there are advantages, traditional DevOps environments are heavily scripted with prescriptive automation scripts, static workflow rules and manual decision making during operations [14]. These rule-based approaches are not well suited for handling large amounts of operational data, for accurate prediction of deployment risks, optimizing infrastructure resources, early detection of failures and adapting deployment strategies when runtime conditions change [15] [16]. These constraints only make it more critical to have intelligent and adaptive software delivery methods.

The predictive analytics [17] [18] capabilities of AI and ML, along with their autonomous decision making and continuous learning capabilities, can turn traditional DevOps into smart software delivery ecosystem. These historical and real-time data are fed to ML algorithms to identify unknown operational patterns, predict deployment failure, estimate deployment risk, detect anomalies, optimize resource usage, suggest rollback plans, and automate root cause analysis before deployments fail in production [19] [20]. Moreover, AI-based decision engines constantly self-learn based on the results of deployment, optimizing CI/CD pipelines, intelligent workload scheduling, proactive infrastructure management and self-healing software delivery. The study's main contributions are:

- Created an end-to-end DevOps automation framework using AI/ML.
- Combined and unified heterogeneous DevOps data from build, monitoring, deployment, and configuration repositories in a learning system.
- Incorporated an intelligent prediction model based on the RF algorithm for deployment risk assessment and decision support.
- Built an AI-driven decision engine for automated deployment optimization, resource allocation, rollback management and continuous monitoring.
- Achieved significant improvements in build efficiency, deploy reliability, predictive accuracy and operational stability versus the traditional DevOps workflow.
- Developed a closed-loop intelligent feedback loop for optimizing and adapting software delivery continuously, continuously improving ML models.

a. Novelty of the paper

The novelty of this work is to provide a unified architecture for the integration of DevOps automation with AI/ML covering several key components, including predictive analytics, intelligent decision-making, automated deployment optimization, rollback management, and continuous learning. The proposed framework is different from the current research, which is limited to specific stages of DevOps, as it sets up an end-to-end intelligent software delivery pipeline with closed-loop feedback and RF-based deployment risk prediction. This integration makes it possible to optimize adaptively, deploy more reliably, minimize operational failures and continuously improve performance across the entire DevOps lifecycle.

B. Structure of the Paper

This is the outline for the remainder of the paper. Section II presents a comprehensive literature assessment. Section III explains the proposed approach. Section IV displays the experimental findings and performance analysis. The study is concluded and future research areas and enhancements are outlined in Section V.

II. Literature Review

While AI in DevSecOps security automation and optimizing CI/CD is well explored, predictive decision-making, self-executing orchestration and adaptive

optimization in the context of comprehensive intelligent DevOps automation is under-explored.

R. R. Mittana et al. (2026), incorporates the threat modeling, secure design synthesis, AI-augmented coding, automated test, infrastructure hardening and adaptive monitoring in the CI/CD pipeline. Two web application prototypes were used for experimental evaluation and revealed an accuracy of 96.7% for vulnerability detection, a false positive rate of 3.8% and an efficiency of 92.4% for application remediation process, compared to five well-established SDL methodologies. The results prove that the framework can deliver continuous, predictive and compliance-based security without jeopardizing the development velocity [21].

R. K. Mahimalur et al. (2025) propose a security-conscious, AI/ML-enhanced DevSecOps approach that incorporates security safeguards at every stage of the software development life cycle to solve these problems. The primary aim is to provide an intelligent, automated, and auditable CI/CD pipeline with careful operational capability for active vulnerability detection and real-time threat mitigation. A GitOps-controlled deployment platform, it integrates runtime policy enforcement, scanning of container images, SCA, SAST, and DAST. AI and ML anomaly detection algorithms are trained on the UNSW-NB15 dataset to accurately identify fraudulent activities. The suggested pipeline outperforms five cutting-edge approaches on operational and security metrics in terms of pre (97.2%), rec (97.5%), acc (97.8%), and latency overhead (3.8) [22].

V. Kalluru (2025) investigates how DevOps pipelines can be automated using IaC principles, and how software-based infrastructure definition and management can streamline software development and deployment. The results show that IaC can help reduce silos, enhance the efficiency and reliability of CI/CD, and foster collaboration between development and operations teams. This study investigates how to improve software delivery workflows by using IaC on a practical level to automate various stages of the DevOps lifecycle [23].

N. G. Camacho (2024) explores effective strategies and practices of leveraging AI and ML to the maximum within DevSecOps environment. The article begins with a general introduction to DevSecOps concepts and the role of AI/ML. It then delves into more specific subjects including automated threat detection, predictive vulnerability management analytics and intelligent automation for continuous integration and deployment. Additionally, it covers important issues like algorithm transparency, data security, and ethical issues when integrating AI/ML into DevSecOps. The paper demonstrates how companies can use AI/ML technology to optimize their DevSecOps pipelines, minimize security risks, and foster a culture of continuous improvement via engaging case studies and real-world analogies [24].

V. H. Das et al. (2024) demonstrate the growing need for simplified solutions. DevOps teams able to simplify company operations by using AI- and ML-driven solutions, freeing up resources for more strategic methods like fostering innovation, tackling difficult problems, and making data-driven choices. The research investigate specific areas where AI and ML can enhance DevOps practices to create a safer, more efficient, and more agile software development environment [25].

Y. Ramaswamy (2023) examines a comprehensive DevSecOps system with security automation embedded in the CI/CD process. Some of the contributions include using policy-as-code tools to enforce policies during runtime, automating static and dynamic analysis, validating IaC compliance, scanning for shift-left security vulnerabilities, and identifying secrets. Jenkins, GitLab CI, SonarQube, Trivy, HashiCorp Vault, and OPA are some of the open-source technologies that they use to build their reference architecture. Evaluate the framework in a simulated agile team environment by looking at metrics like ScanLatency, FP, vulnerability DetectionRates, and MTTR. The findings show improved pipeline traceability and a notable decrease in post-deployment vulnerabilities [26].

Table I summarizes recent studies on AI/ML-driven DevOps automation by comparing applied AI techniques, autonomous functions, key findings, identified limitations, and suggested future research directions.

TABLE I. LITERATURE COMPARISON OF AI/ML-DRIVEN DEVOPS AUTOMATION FRAMEWORKS

Authors	AI Technique	Autonomous Function	Key Findings	Limitations	Future Work
R. R. Mittana et al. (2026)	AI-augmented coding, predictive vulnerability detection	Threat modeling, automated testing, adaptive monitoring	Achieved 96.7% vulnerability detection accuracy, 3.8% false positives, and 92.4% remediation efficiency.	Evaluated only on two web applications; enterprise scalability not validated.	Extend to large-scale, multi-cloud DevSecOps with adaptive learning.
R. K. Mahimalur et al. (2025)	AI/ML anomaly detection (UNSW-NB15)	Automated vulnerability detection and CI/CD security	Achieved 97.8% accuracy, 97.2% precision, 97.5% recall, and 3.8% latency overhead.	Relies on a single dataset and focuses mainly on security automation.	Explore multi-dataset learning and intelligent deployment optimization.
V. Kalluru (2025)	IaC-based automation (limited AI)	Infrastructure provisioning and pipeline automation	IaC improved collaboration, deployment consistency, and CI/CD efficiency.	Lacks AI-driven decision-making and predictive automation.	Integrate AI for infrastructure optimization and autonomous configuration.
N. G. Camacho (2024)	AI/ML-based predictive analytics	Threat detection and security automation	AI/ML improved threat detection, predictive security, and DevSecOps automation.	Conceptual study with limited experimental validation.	Develop validated AI frameworks using real-world DevSecOps datasets.
V. H. Das et al. (2024)	AI/ML-assisted DevOps	Process automation and decision support	AI/ML enhanced software delivery efficiency, agility, and security.	Limited implementation details and performance evaluation.	Validate AI-enabled DevOps frameworks in industrial environments.
Y. Ramaswamy (2023)	AI-assisted security automation	Security scanning, IaC validation, policy enforcement	Reduced post-deployment vulnerabilities and improved pipeline traceability.	Focuses on security automation with limited predictive intelligence.	Incorporate ML for autonomous remediation and predictive CI/CD optimization.

III. METHODOLOGY

The proposed framework designed to enhance software delivery through intelligent automation and continuous optimization as shown in Figure. 1. The process begins with data collection, where operational, application, and infrastructure data are gathered from multiple sources. This is followed by data pre-processing and integration to clean, standardize and consolidate the data for analysis. Then, feature engineering is used to find useful attributes that boost model performance, followed by splitting the data into training and testing data using dataset splitting.

The prepared data is then used for the creation of AI/ML models, which can make predictive and adaptive decision-making possible. This trained model feeds into an AI-based decision engine that can be used to automate deployment via a CI/CD pipeline. After deployment, the system checks if the deployment has succeeded. If unsuccessful, a built-in auto-rollback and recovery feature re-stabilizes system. Successful deployments further execute performance evaluation to capture results, leading to continuous improvement, reliability, resilience and efficient software delivery.

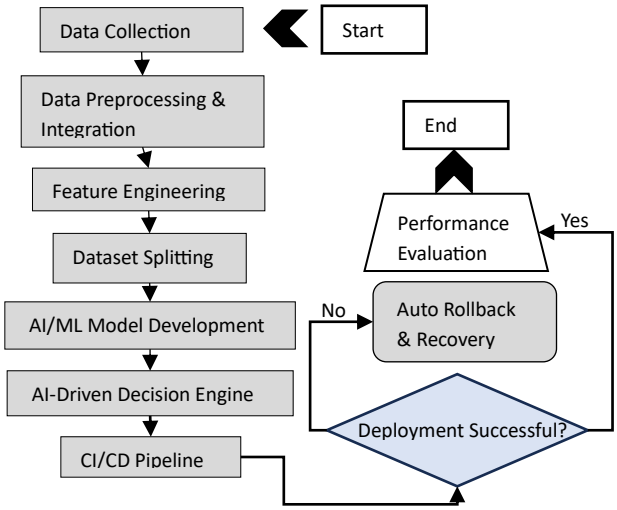

Fig. 1. Proposed Framework for Intelligent Software Delivery and Continuous Deployment

a. Data Collection

Historical DevOps data was gathered for six months from build logs and performance metrics, deployment history and deploy success, configuration repository and change logs, system monitoring, alerting platforms, incident reports and resolution history, and user feedback logs. The data sources were heterogeneous, and they were synchronized using a pipeline identifier and timestamps to create a unified dataset for training and evaluating the proposed ML models.

B. Data Preprocessing

Data collected during DevOps were preprocessed to improve their quality, consistency and reliability. This process included the following steps:

- **Log Cleaning:** Duplicate, incomplete and corrupted data from build logs, deployment histories, monitoring data, configuration repositories, and incident reports, ensuring data consistency.
- **Timestamp Synchronization:** Consistent timestamps in all DevOps tools to ensure that time is consistent and events are synced with the software delivery pipeline.
- **Missing Value Treatment:** The matching mode was used to replace the missing values for category variables and the median value for numeric variables.
- **Log Parsing and Feature Extraction:** Analysed raw logs and calculated operational metrics like build time, deployment success, resource usage, failure percentage, configuration changes, incident severity, and customers feedback.
- **Categorical Encoding:** One-hot encoded categorical attributes to use numerical values for training ML models.

C. Feature Scaling

Normalized continuous features are numeric features that are brought into a range of values between 0 and 1. This reduces the effect of feature size, helps it converge during training and improves the ML model prediction. Min-Max normalization is carried out using the equation (1):

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (1)$$

D. Data Integration

The log files of builds, deployment track information, monitoring metrics, configuration change logs, incident histories, etc., and user feedback were pre-processed and merged into a single dataset, using common pipeline identifiers and synchronized times. The integration connected various stages of the DevOps pipeline, ensuring data consistency, avoiding data duplication, and gaining a holistic perspective of the software delivery lifecycle for precise training and evaluation of the ML model.

E. Feature Engineering

The integrated data set was mapped to a structured feature space that can be exploited by ML. Feature engineering included the operational, temporal and historical characteristics of the DevOps pipeline to produce informative predictors that recapture the behaviour of execution and the dynamics of deployment to boost the discriminative power of the learning model.

F. Dataset Partitioning

The integrated dataset was split into three: 70% were used to train the model, 15% were used to validate the model, and 15% were used to test the model and evaluate its ultimate performance.

G. Random Forest Model

The algorithm used is RF, which was selected as one of the most representative ML algorithms due to its ability to deal with high dimensional data and multiple types of data, as well as to obtain a high prediction accuracy that is also highly resistant to overfitting [27]. RF is an Ensemble Learning method supervised by training many DTs from randomly selected samples from training data and using the input attributes. The results of individual trees can be combined by voting for classification or averaging for regression, enhancing prediction accuracy and stability of the model. It is useful for predictive analytics and classification problems due to its ability to handle complex nonlinear relationships. The model is built by stacking outputs of multiple DTs, which can reduce the variance, enhance the generalization ability, and generate stable predictions for unseen data. The hyperparameters were fine-tuned using the validation subset, while the RF model was trained using the training subset. The RF model's prediction can be stated as follows Equation (2)

$$\hat{y} = \text{mode}\{T_1(x), T_2(x), \dots, T_n(x)\} \quad (2)$$

where $T_n(x)$ represents the decision tree's prediction, n represents the sum of all decision trees, x depicts the feature vector input., and $\hat{y}$ is the sum of all trees' forecasts used as the final output.

H. AI-Driven DevOps Decision Engine

The decision engine performs intelligent build optimization, predictive deployment risk assessment, dynamic resource allocation, intelligent cache management, rollback recommendation, and automated alert generation. Based on the predicted risk scores and system conditions, the engine recommends optimal execution strategies while minimizing manual intervention. This stage serves as the central intelligence layer that continuously guides software delivery decisions.

I. Continuous Integration and Deployment Pipeline

The optimized DevOps workflow proceeds through the CI/CD pipeline. Every source code commit initiates an AI-assisted build process followed by automated testing and pre-deployment risk assessment. If the deployment risk estimation is still lower than the set limit, the application is deployed into the production environment, and runtime performance data are collected through continuous monitoring. This automatic process allows problems with software to be identified in advance, before they impact production systems.

J. Rollback Management

The framework proposed includes an intelligent rollback mechanism, which reduces service disruption during software deployment. Monitoring health through real-time health checks is continuously evaluated by AI-based risk predictions and operational metrics. The system then begins to roll back to the last stable release automatically or by operator action, if the deployment risk is above the specified acceptance level. After rollback, the integrity and stability of the deployment are checked prior to restoring normal operations. All rollback events, associated risk scores, and recovery outcomes are stored for future model retraining to improve the reliability of future deployments and decrease the frequency of rollbacks.

K. Performance Metrics

The proposed AI/ML based DevOps framework tested by comparing the traditional DevOps pipeline with the DevOps pipeline using AI. These metrics capture the impact AI/ML has on build efficiency, resource usage, reliability of deployment, and rollback management.

Average Build Time (ABT): This is the typical time needed to develop the program. Faster build times translates to more efficient pipelines and faster software delivery. It is calculated as shown in (3):

$$ABT = \frac{\sum_{i=1}^{N} BT_i}{N} \quad (3)$$

Build Failure Rate (BFR): The percentage of failed software builds while running them. Lower values are better for build reliability (Equation (4)).

$$BFR = \frac{\text{Number of Failed Builds}}{\text{Total Number of Builds}} \times 100 \quad (4)$$

Deployment Failure Rate (DFR): Deployment Failure Rate is the ratio of deployment failures to deployments during the deployment of the software. Smaller deployment failures indicate deployment stability as shown in equation (5).

$$DFR = \frac{\text{Failed } Deployments}{\text{Total Deployment Attempts}} \times 100 \quad (5)$$

Accuracy: It represents the ability of the AI/ML model to correctly classify deployment risk. The more accurate, the more reliable the deployment decision making process. It is derived from the following equation:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \times 100 \quad (6)$$

Rollback Frequency (RF): Rollback Frequency is the ratio of the number of deployments that needed to be rolled back for failures or instability. The lower rollback frequency is understood as the better quality of the deployment, according to the Equation (7).

$$RF = \frac{\text{Number of } Rollbacks}{\text{Total Deployment Attempts}} \times 100 \quad (7)$$

These metrics assess the proposed AI/ML framework as a whole. Build optimizations and deployment performance are significantly improved as shown in the evaluation.

IV. Result Analysis and Discussion

The success of the proposed DevOps automation framework using AI/M.ths. is evaluated by the key performance indicators related to build efficiency, deployment reliability and risk management. The outcomes produced are analyzed and compared with the existing literature from a theoretical perspective and the improvements are brought by the framework in the field of intelligent automation, operational efficiency and continuous software delivery. Table II shows the impact of AI/ML techniques on software build process in DevOps environments. The results show that significant benefits have been achieved, with an average reduction of approximately 50% in build time, higher resource utilisation, reduced build failures and increased cache efficiency. The enhancements show quicker, more dependable, and scarce resource software development processes.

TABLE II. Impact of AI/ML on Build Process Performance in DevOps Pipelines

Metric	Baseline	With AI/ML	Improvement
Average Build Time	45 min	22.5 min	50%
Resource Utilization	60%	75%	25%
Build Failure Rate	8%	4.8%	40%
Cache Hit Rate	65%	91%	40%

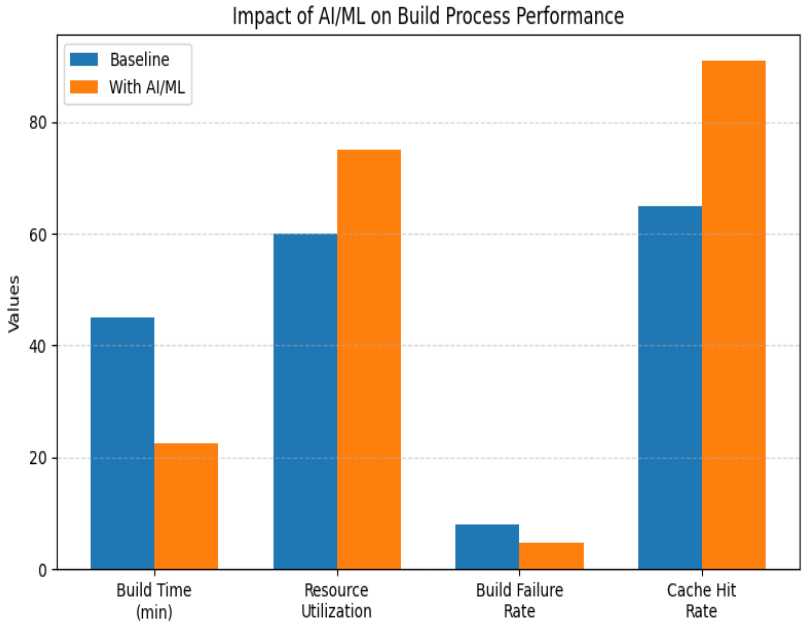

Fig. 2. Impact of AI/ML on Build Process Performance in DevOps Pipelines

This is the difference between AI/ML enabled build processes in DevOps pipelines and non-AI/ML enabled ones (see Fig. 2). The graph shows that significant improvements have been achieved, such as lower build times, better resource usage, fewer build failures and better cache hit rates. Results demonstrate the potential advantages of AI/ML-powered automation in boosting build efficiency, resource optimization, and software development process stability.

The usefulness of AI/ML for improving deployment reliability and operational risk management can be summarized in Tab III. Automation reduces deployment failures, rollbacks and false alarms, and improves prediction accuracy with AI-powered automation. The results of this study demonstrate the benefits of intelligent deployment strategies to enhance software stability, predict and mitigate risks during operation, and boost the performance of continuous delivery.

TABLE III. AI/ML-BASED DEPLOYMENT RELIABILITY AND RISK MANAGEMENT PERFORMANCE

Metric	Baseline	With AI/ML	Improvement
Deployment Failure Rate	15%	6%	60%
Risk Prediction Accuracy	65%	87%	34%
Rollback Frequency	12%	6%	50%
False Positive Rate	25%	8%	68%

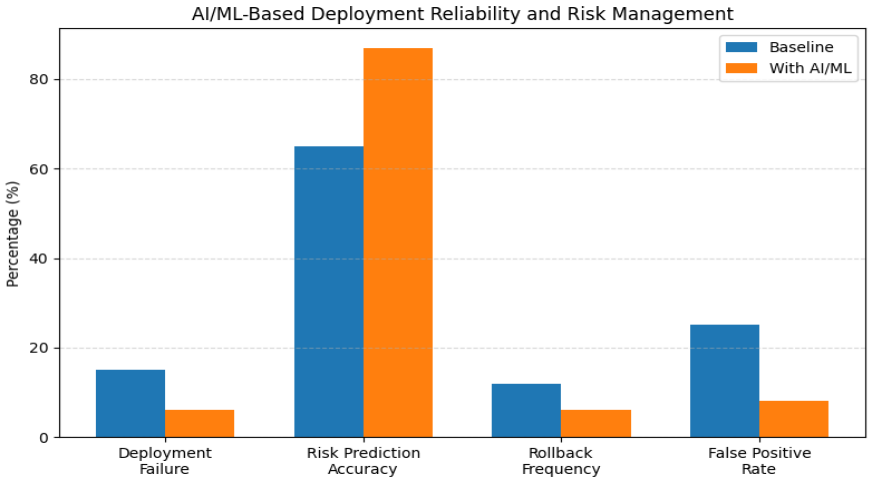

Fig. 3.AI/ML-Based Deployment Reliability and Risk Management Performance

The baseline performance is shown as the reference for calculating how AI/ML impacts deployment reliability and risk management, as depicted in Fig. 3. The results show that the rate of failure on deployment is decreased, rollback frequency is decreased, the risk prediction accuracy is improved, and the number of FP is reduced. The improvements point to the impact that intelligent automation can have on deployment stability, operational risk reduction and reliable continuous software delivery.

- *Comparative Analysis & Discussion*

Table IV compares the proposed AI/ML-based DevOps automation framework with three recent studies in a theoretical fashion. The differences between the two areas of research are emphasized, as is the automation degree, AI/ML application, feedback loop, and contribution overall. The proposed framework provides for comprehensive lifecycle automation, based on intelligent decision-making and ongoing performance optimization, in contrast to the approaches currently offered in the market, which are specific for certain phases of the DevOps lifecycle.

The experiment results show that the use of AI/ML has a positive impact on software delivery performance for various operational metrics when it is used in DevOps. The proposed framework greatly reduces the deployment failure rate, false alarms, rollback frequency and the deployment time required, and by intelligent decision-making, prediction accuracy of deployment risk is improved. The RF model is well suited to modelling complex patterns of operation when data is heterogeneous from DevOps, making it possible to predict deployment risks before product deployment. Additionally, the Artificial Intelligence-based decision engine enables smart resource usage, automatic rollback suggestions and on-demand performance optimization without too much manual effort. In summary, the proposed framework provides greater lifecycle automation and adaptive feedback as compared to existing approaches that boost software reliability, continuity of operations, deployment uniformity, and the effectiveness of the CI/CD pipeline in the modern software development environment.

TABLE IV. COMPARISON OF THE PROPOSED AI/ML-DRIVEN DEVOPS AUTOMATION FRAMEWORK WITH EXISTING STUDIES

Comparison Aspect	[28]	[29]	[30]	**Proposed Framework**
Research Focus	Applies ML for predictive failure detection and adaptive deployment.	Optimizes CI/CD pipelines using AI-driven automation.	Improves deployment resilience through	Integrates AI/ML across the complete DevOps lifecycle for intelligent software delivery.

			self-healing mechanisms.	
Automation Scope	Focuses on deployment prediction and rollback.	Automates CI/CD workflows and deployment processes.	Emphasizes automated recovery after deployment failures.	Automates data processing, model development, deployment, monitoring, recovery, and evaluation.
AI/ML Integration	ML predicts deployment failures before release.	AI enhances deployment scheduling and pipeline execution.	ML detects anomalies and triggers recovery actions.	AI/ML supports prediction, intelligent decision-making, deployment optimization, and adaptive recovery.
Feedback Mechanism	Limited post-deployment learning.	Performance monitoring for deployment evaluation.	Feedback loops enable automated self-healing.	Closed-loop feedback continuously refines AI models and deployment decisions.
Research Contribution	Enhances deployment reliability through predictive analytics.	Improves CI/CD efficiency using AI-based optimization.	Strengthens system resilience through automated recovery.	Provides a unified intelligent DevOps framework combining predictive analytics, autonomous automation, and continuous optimization.

V. Conclusion and Future Scope

AI and ML can be used to enhance DevOps practices with intelligence, adaptability, and reliability, meeting the needs of today's complex software development. The framework will be rolled into a single DevOps architecture, along with predictive deployment risk analysis, intelligent resource allocation, automated rollback management and continuous performance optimization. By taking advantage of a prediction model built using the RF technique, the framework gathers various operation data to allow proactive deployments and to keep software reliability, reducing operational failures and manual work. Results from the experiments show that the use of AI-driven automation for continuous software delivery is effective, resulting in significant benefits for build efficiency, deployment stability, rollback reductions, prediction accuracy, and overall CI/CD performance. The closed-loop feedback mechanism is further reinforcing the framework by learning from the results of the deployments and enhancing future decision making processes. Future works will investigate advanced DL and RL algorithms for optimizing deployment autonomously, large language models for improving the software delivery capabilities of intelligent DevOps assistants, extension to multi-cloud, edge and hybrid computing scenarios, integration of XAI techniques for ensuring explanation of the recommendations made during the deployment, and validation of the approach with large industrial datasets and real streaming infrastructures to improve scalability, robustness, cybersecurity, and full autonomous software delivery.

References

1. S. Chatterjee, “A Data Governance Framework for Big Data Pipelines: Integrating Privacy, Security, and Quality in Multitenant Cloud Environments,” *Tech. Int. J. Eng. Res.*, vol. 10, no. 5, 2023, doi: 10.56975/tijer.v10i5.158181.
2. S. Jain and D. Jain, “Artifact Comparison Analyzer: Evaluating Microservice Build Metrics for Performance and Efficiency Improvements,” in *2026 IEEE International Conference on AI Engineering and Innovations (AIEI)*, Jamshedpur, India: IEEE, Mar. 2026, pp. 1–6. doi: 10.1109/AIEI69164.2026.11497468.
3. V. Sharma, “Cloud-Native 5G Deployments: Kubernetes and Microservices in Telco Networks,”

Int. J. Innov. Res. Eng. Multidiscip. Phys. Sci., vol. 10, no. 3, pp. 1–8, May 2022, doi: 10.37082/IJIRMPS.v10.i3.232706.

4. C. Shekhar Pareek, "Chaos Testing: A Proactive Framework for System Resilience in Distributed Architectures," *Int. J. Sci. Res.*, vol. 13, no. 11, pp. 851–855, Nov. 2024, doi: 10.21275/SR241110081650.
5. S. R. Chanthati, "Implementing a Graph Neural Network (GNN) in Amazon Web Services (AWS) involves setting up infrastructure for data processing, training, and deployment," in *Proceedings of the 6th World Conference on Artificial Intelligence: Advances and Applications (WCAIAA 2025)*, May 2025. doi: 10.5281/zenodo.20313312.
6. M. Mittal, "The Great Migration: Understanding the Cloud Revolution in IT," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 10, no. 6, pp. 2222–2228, Dec. 2024, doi: 10.32628/CSEIT2410612423.
7. L. A. Yeruva, D. Singh, S. Suddala, N. Bhatt, and R. Uddin, "Augmented Data Management for Cache Performance, Cybersecurity, and Mobile Integration," *J. Comput. Mech. Manag.*, vol. 5, no. 3, Jun. 2026, doi: 10.57159/jcmm.5.3.26691.
8. R. Azmeera and J. Hyatt, "Software Errors, Product Functionality, and Organizational Cascades: Evidence from Developer-Lived Experience," Mar. 2026. doi: 10.2139/ssrn.6924439.
9. J. B. Mehta, "Designing Self-Healing Automation Frameworks for Flaky CI Environments," in *2025 International Conference on Computer and Applications (ICCA)*, IEEE, Dec. 2025, pp. 1–7. doi: 10.1109/ICCA66035.2025.11430985.
10. M. R. C. Mukkolakkal, "Deploy, Calibrate, Monitor, Heal -- No Human Required: An Autonomous AI SRE Agent for Elasticsearch," *arXiv.org*, Apr. 2026.
11. R. K. Kanneganti, "Security and Compliance Issues in Cloud-Based Deployments of Content and Workflow Management Systems," *Eastasouth J. Inf. Syst. Comput. Sci.*, vol. 2, no. 01, pp. 131–138, Aug. 2024, doi: 10.58812/esiscs.v2i01.1122.
12. H. P. Cyril, N. D. Bhandarwar, S. Kumara, and S. Mathur, "Securing Containerized Applications Using Kubernetes-Based Orchestration in Software Development," in *2026 International Conference on Intelligent Computing, Networks, and Security (IC-ICNS)*, 2026, pp. 1–6. doi: 10.1109/IC-ICNS68863.2026.11537887.
13. V. Chaturvedi, "Modern Software Development with Java , Spring Boot , and Python : A Survey of Frameworks and Best Practices," *ESP J. Eng. Technol. Adv.*, vol. 3, no. 4, pp. 188–197, 2023, doi: 10.56472/25832646/JETA-V3I8P121.
14. K. K. Mohammed, "Leadership Practices of Data Engineering for AI and Machine Learning," *Int. J. Sci. Res. Eng. Trends*, vol. 12, no. 1, 2026, doi: 10.5281/zenodo.18677279.
15. R. Palwe, "Onboarding for AI features: Reducing friction at the first use," *Int. J. Comput. Artif. Intell.*, vol. 6, no. 2, pp. 393–400, Jul. 2025, doi: 10.33545/27076571.2025.v6.i2e.227.
16. R. Vasikarla, S. Kakkar, B. Makkena, and A. Kakkar, "An Enhanced Artificial Intelligence Framework for Cloud Resource Usage Anomaly Detection in Site Reliability Engineering," in *2026 2nd International Conference on Cognitive Computing in Engineering, Communications, Sciences and Biomedical Health Informatics (IC3ECSBHI)*, IEEE, Feb. 2026, pp. 1418–1424. doi: 10.1109/IC3ECSBHI67834.2026.11468993.
17. R. Lingam, "Integrating Trustworthiness Into the AI Lifecycle (Trustworthiness)," in *AI Safety and Preventing Harm in AI Systems*, IGI Global Scientific Publishing, 2026, pp. 213–248. doi: 10.4018/979-8-3373-6935-8.ch008.
18. S. K. Malaraju and S. K. Madishetty, "AI-Augmented Compiler Optimization for Energy-efficient Software Execution on Embedded Systems," in *2025 14th International Conference on System Modeling & Advancement in Research Trends (SMART)*, Moradabad, Uttar Pradesh, India: IEEE, 2025, pp. 1–6, November. doi: 10.1109/SMART66937.2025.11389313.
19. S. K. Anumula, "AI-Powered Cybersecurity Framework for Cloud-Based Applications: Enhancing malware detection and threat response using deep learning," *Int. J. Syst. Des. Inf. Process.*, vol. 13, no. 4, pp. 109–116, 2025, doi: 10.64971/j.cph.ijsdip.v13.14.16.2025.
20. A. Joon, B. K. R. Janumpally, A. Gogineni, and P. Chatterjee, "Efficient Large-Scale Intrusion Identification and Prevention in Distributed Cloud Networks Using Artificial Intelligence," in *2025 5th International Conference on Intelligent Technologies (CONIT)*, 2025, pp. 1–8. doi: 10.1109/CONIT65521.2025.11167760.
21. R. R. Mittana, V. Sannamuri, S. Mandru, A. Gunuganti, S. T. Meesala, and A. A. Syed, "Secure Development Lifecycle in Web Applications:

Bridging the Gap Between Development and Security," in *2026 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*, 2026, pp. 1–9. doi: 10.1109/ICSES66558.2026.11479021.

22. R. K. Mahimalur, S. Amgothu, B. R. T. Reddy, and S. S. Gadde, "Modern Cloud Security and Automation: A DevSecOps Approach Leveraging AI/ML and Containerization," in *2025 9th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, IEEE, Nov. 2025, pp. 305–312. doi: 10.1109/ICECA66444.2025.11383338.

23. V. Kalluru, "Declarative Automation of DevOps Workflows through Infrastructure as Code," Nov. 2025. doi: 10.22541/au.176236358.85778824/v1.

24. N. G. Camacho, "Unlocking the potential of AI/ML in DevSecOps: effective strategies and optimal practices," *J. Artif. Intell. Gen. Sci.*, vol. 2, no. 1, 2024.

25. V. H. Das Chowdary, A. Shanmukh, T. P. Nikhil, B. S. Kumar, and F. Khan, "DevOps 2.0: Embracing AI/ML, Cloud-Native Development, and a Culture of Continuous Transformation," in *2024 4th International Conference on Pervasive Computing and Social Networking (ICPCSN)*, IEEE, May 2024, pp. 673–679. doi: 10.1109/ICPCSN62568.2024.00112.

26. Y. Ramaswamy, "DevSecOps in Practice: Embedding Security Automation into Agile Software Delivery Pipelines," *J. Comput. Anal. Appl.*, vol. 31, no. 4, 2023, doi: 10.48047/jocaaa.2023.31.04.27.

27. K. Jangiti, "Design and Validation of a Machine Identity Governance Framework for AI Agents in Multi-Cloud Environments," in *SoutheastCon 2026*, IEEE, Feb. 2026, pp. 1–6. doi: 10.1109/SoutheastCon63549.2026.11476363.

28. V. M. Tamanampudi, "AI-Enhanced Continuous Integration and Continuous Deployment Pipelines: Leveraging Machine Learning Models for Predictive Failure Detection, Automated Rollbacks, and Adaptive Deployment Strategies in Agile Software Development," *Distrib. Learn. Broad Appl. Sci. Res.*, vol. 10, pp. 56–96, 2024.

29. R. Kakarla and S. B. Sannareddy, "AI-Driven DevOps Automation for Ci/Cd Pipeline Optimization," *Eastasouth J. Inf. Syst. Comput. Sci.*, vol. 2, no. 01, pp. 70–78, Aug. 2024, doi: 10.58812/esiscs.v2i01.849.

30. A. Challa, "Self-Healing CI/CD Pipelines with Feedback-Loop Automation: Building Fault-Tolerant CI/CD Systems Using Anomaly Detection and Automated Rollback Logic," *Int. J. Intell. Syst. Appl. Eng.*, vol. 12, no. 23s, p. 3217, 2024.