INTEGRATION OF MODERN ARTIFICIAL INTELLIGENCE TECHNOLOGIES IN THE PROCESSES OF CONTINUOUS INTEGRATION AND DEPLOYMENT OF SOFTWARE

Adrian Nakonechny, Dr. Sc, professor, Mykhail-Mishel Vyhrynovskyi, PhD Candidate,
National University "Lviv Polytechnic", Ukraine
e-mail: mykhail-mishel.a.vyhrynovskyi@lpnu.ua

https://doi.org/10.23939/istcmtm2025.03.

Abstract. This article discusses modern approaches to organizing continuous integration (CI) and continuous delivery (CD) processes in software development using artificial intelligence (AI) technologies. The historical development of CI/CD is analyzed, along with their role in ensuring high-quality software, the main advantages and disadvantages of traditional approaches, and the prospects for integrating AI technologies to automate and optimize these processes. The research results demonstrate that a comprehensive implementation of CI/CD systems utilizing AI contributes to shorter development cycles, increased system stability, and more efficient resource usage.

Key words: CI/CD, continuous integration, continuous delivery, artificial intelligence, automation, optimization, DevOps.

1. Introduction

In recent decades, software development methods have undergone radical changes. Traditional waterfall models have gradually receded into the background, while modern approaches, particularly continuous integration (CI) and continuous delivery (CD), have become the foundation of efficient development team operations. The contemporary development model based on DevOps (development and operations) principles provides close integration between development teams and operations specialists, enabling the rapid deployment of new functionalities and the maintenance of stable application performance in production environments. Modern AI technologies open additional possibilities for optimizing CI/CD processes, leading to high-quality end products [1-6]. This article presents an analysis of modern approaches to CI/CD implementation, identifies their strengths and weaknesses, and examines the prospects for applying artificial intelligence (AI) to automate and optimize these processes, with an emphasis on practical technologies and examples of their use.

2. Drawbacks of Modern Approaches

Despite the numerous advantages of implementing CI/CD, there are certain challenges that complicate their full-scale application in large information systems.

1. Integration of Cutting-Edge Technologies.
2. Implementing innovative solutions, in particular AI technologies, requires significant financial and human resources, as well as the modernization of existing infrastructure. This can become a serious barrier for small and medium-sized companies.
3. Processing Large Volumes of Data.
4. Predicting defects and optimizing test scenarios require the analysis of large volumes of historical data, which is associated with high computational power demands and the use of specialized algorithms.
5. Lack of Standardization. The variety of approaches and solutions used by different companies creates difficulties in implementing unified standards, complicating the integration of innovative technologies into existing CI/CD pipelines.

In order to overcome these challenges, it is necessary to develop methodologies that ensure the integration of modern technologies into traditional CI/CD systems, optimize data processing methods, and implement standards [1,2].

3. Research Objective

The main objective of this research is to develop and substantiate a methodology for integrating artificial intelligence technologies into the processes of continuous integration and deployment that will: increase the efficiency and stability of software development processes; reduce the response time to emerging defects through automated prediction and optimization of test scenarios; optimize system resource usage through adaptive real-time load management.

Ensure the integration of various automation tools into a unified ecosystem using modern AI technologies.

4. Research on Modern Methods for Automating and Optimizing CI/CD Processes Using Artificial Intelligence

In a modern IT environment, continuous integration and delivery (CI/CD) have become key components of the software development lifecycle, allowing teams to rapidly introduce innovations and maintain a high product quality level. Traditional development methods are gradually taking a back seat, while modern approaches based on DevOps principles enable the creation of seamless pipelines for building, testing, and deployment.

However, there are many challenges associated with implementing CI/CD in large systems: the need for significant financial and human resources to integrate advanced technologies, the processing of large volumes of historical data with high computational demands, and the lack of unified standards for implementing innovative solutions.

The use of modern DevOps approaches fosters deep automation, and CI/CD systems can be greatly enhanced by integrating AI technologies. An analysis of advanced DevOps practices confirms the necessity of improving processes through the implementation of artificial intelligence, which increases the efficiency and reliability of CI/CD pipelines [7]. Further studies highlight how AI can enhance the security and reliability of CI/CD by detecting anomalies and automating incident responses [8]. The integration of AI into CI/CD pipelines opens up the following prospects: algorithms such as recurrent neural networks can analyze historical data on code changes and test outcomes, forming a defect prediction model mathematically described as $P(defect) = f(x_1, x_2, \ldots, x_n)$, where $x_1, x_2, \ldots, x_n$ represent the characteristics of the changes and $f( )$ is a function built on the basis of machine learning [9]. This approach allows for the early identification of potential problem areas, optimizing the allocation of resources and reducing the time required for defect resolution. In addition, optimizing test scenarios using AI through test case prioritization methods based on machine learning significantly accelerates the process of verifying critical application components. Finally, optimizing continuous delivery using machine learning methods, particularly reinforcement learning algorithms for adaptive deployment management, demonstrates AI's potential for improving both the stability and quality of the final product [10].

Containerization, provided by the management toolkit for isolated Linux containers Docker, is fundamental for modern CI/CD pipelines. With Docker, it is possible to create single, reproducible application images containing all necessary dependencies, ensuring environmental compatibility across all stages of development. The Dockerfile defines a base image, sets a working directory, copies dependency files and the application's source code, installs dependencies, opens the port, and launches the application. This allows for the effective integration of AI modules that analyze test results in containers and provide recommendations for configuration optimization [12].

To manage a large number of containers, the open-source Kubernetes system is used, which ensures automatic scaling and application fault tolerance. Thanks to the rolling updates deployment strategy, Kubernetes allows for updates without system downtime. The integration of AI with Kubernetes, particularly through the analysis of monitoring data, optimizes scaling configurations and increases resource management efficiency. Additionally, ArgoCD as a modern declarative deployment tool ensures synchronization of the actual state of applications with the desired configuration specified in YAML files. The use of AI for analyzing deviations in system states helps to react promptly to potential problems and maintains consistent adherence to the desired state [13].

For automating CI/CD processes, orchestration systems such as Jenkins are often utilized. The Jenkins pipeline configuration can be described using a Jenkinsfile, which is stored in the code repository. The Jenkinsfile defines the pipeline with three stages: Build, Test, and Deploy. In the Build stage, the project is built using a project build automation tool like Maven; in the Test stage, unit tests are executed; and in the Deploy stage, the application is deployed to Kubernetes via the kubectl command line. Cloud integration through Amazon Web Services (AWS) makes it possible to implement highly efficient CI/CD pipelines with automatic scaling and high security. AWS services, such as CodePipeline, CodeBuild, and CodeDeploy, allow for the automation of the entire development lifecycle, while ECS and EKS services simplify container management in a cloud environment. AI integration with AWS, which provides log analysis and load prediction, contributes to optimal resource allocation and system stability-an essential aspect for ensuring high application quality [14-15].

Within the scope of research on automating CI/CD processes with artificial intelligence, an architecture (see Fig. 1) was developed that illustrates the interaction between the main components of software deployment. This architecture reflects an extended CI/CD infrastructure that includes the integration of ArgoCD, Docker Artifactory, and AI analytics, ensuring a continuous process of updating and optimization.

ArgoCD as a Key Element of GitOps Deployment. ArgoCD is a critical component in the GitOps approach that ensures automatic synchronization of Kubernetes cluster states with the specified configurations. Unlike traditional deployment methods, ArgoCD reduces the need for manual intervention and minimizes the risks of application version mismatches. The main features provided by ArgoCD include:

– Automated application state management - comparing the actual cluster state with the desired state as defined in a Git repository.

– Rollback - in the event of erroneous changes, ArgoCD automatically reverts the system to a stable state.

– Enhanced security and access control - integration with RBAC (Role-Based Access Control) allows for flexible user permissions management.

– Docker Artifactory for Container Image Management**.** Docker Artifactory is used as a centralized repository for container images that pass through the CI/CD pipeline. The main advantages of this approach are:

– Versioning and change control - Artifactory allows tracking of container changes and supports multiple versions.

– Caching and performance optimization - reusing images reduces the load on the CI/CD system.

– Integration with security policies - ensuring automatic vulnerability scans of container images before deployment.

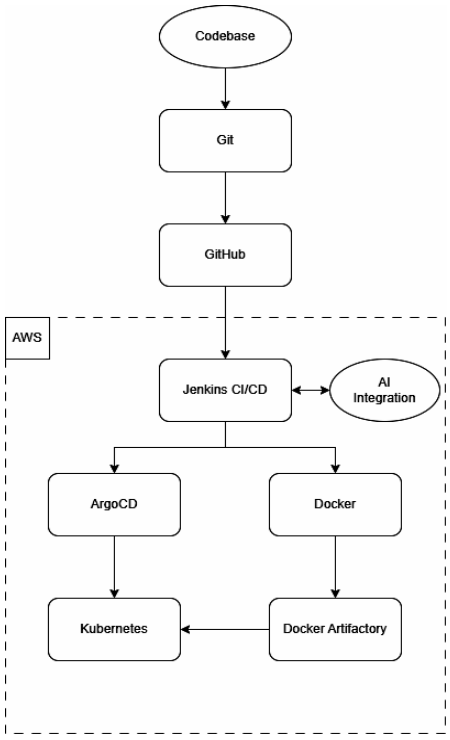

Fig. 1. Architectural scheme of the interaction between the main components of software deployment.

AI Analytics in CI/CD. The architecture includes an AI Integration system that connects to Jenkins CI/CD to analyze deployment processes, which enables:

– Prediction of potential failures based on the analysis of historical deployment data.

– Optimization of resources through dynamic adjustment of container configurations according to actual load.

– Automatic analysis of test errors and suggesting ways to resolve them.

Thus, the proposed architecture demonstrates an advanced approach to DevOps automation, where the combination of ArgoCD, Docker Artifactory, and AI analytics creates an efficient, adaptive, and secure software deployment pipeline.

Within the framework of optimizing CI/CD pipelines, an architectural scheme (see Fig. 2) was developed that describes the detailed sequence (Pipeline) of automated software building, testing, and deployment.

Description of the Architecture. This diagram illustrates the complete cycle of code processing in an automated CI/CD Pipeline, starting from committing changes to the version control system (SCM) and ending with deployment and status notification.

Main Pipeline Stages:

1. SCM Commit identifies changes in the source code management system. The process is automatically initiated once the developer commits changes to the version control system. The choice of branch determines the target deployment environment.

2. Authentication. User verification and authorization of actions are performed to ensure security before proceeding to the following stages.

3. Dependency Download/Install. At this stage, the necessary libraries and modules are downloaded, and a dependency security scan is performed to detect potential vulnerabilities.

4. SCM Checkout. Ensures retrieval of the current state of the repository for further processing.

5. Static Code Analysis. Detects errors and potential vulnerabilities by analyzing the code structure without executing the program.

6. Build. During the build process, parallel compilation and processing of several independent modules (Build Module 1, Build Module N) can take place, which speeds up the process.

7. Artifact Storage. The built artifacts are stored for subsequent stages, including deployment.

8. Test. Involves executing various types of tests in parallel to optimize time:

– Integration Test verifies the interaction between modules.

– Regression Test detects the impact of new changes on existing functionality.

– Smoke Test checks the basic operability of the application.

– Accessibility Test ensures compliance with WCAG standards.

– Performance Test analyzes system speed. The test results are compiled into a Tests Report.

9. Docker Image Scan. Automatic analysis of container images to identify potential security threats.

10. Deploy. Updated services are delivered to the target environment.

11. Post-Deployment Tests/Monitoring. Analyzes performance, collects metrics, and verifies the stability of updates.

12. Notifications, Build Report. The final stage involves generating reports on pipeline execution and notifying teams of the status.

13. End of Workflow. The final point of the process, indicating either successful or unsuccessful deployment completion.

This architectural scheme demonstrates a modern approach to CI/CD process automation that emphasizes time optimization, enhanced code reliability, and improved release quality. The combination of parallel test execution, automated security scanning, and effective artifact storage creates a flexible and scalable continuous delivery system.

In summary, the comprehensive integration of modern CI/CD systems with technologies such as Docker, Kubernetes, ArgoCD, and AWS, augmented with artificial intelligence capabilities, creates a powerful ecosystem capable of ensuring highly efficient automation, optimized testing and deployment processes, and adaptive resource management. This approach not only helps reduce operational costs, but also enables timely defect detection and rapid response to changes in load - a key factor for the success of modern IT systems.

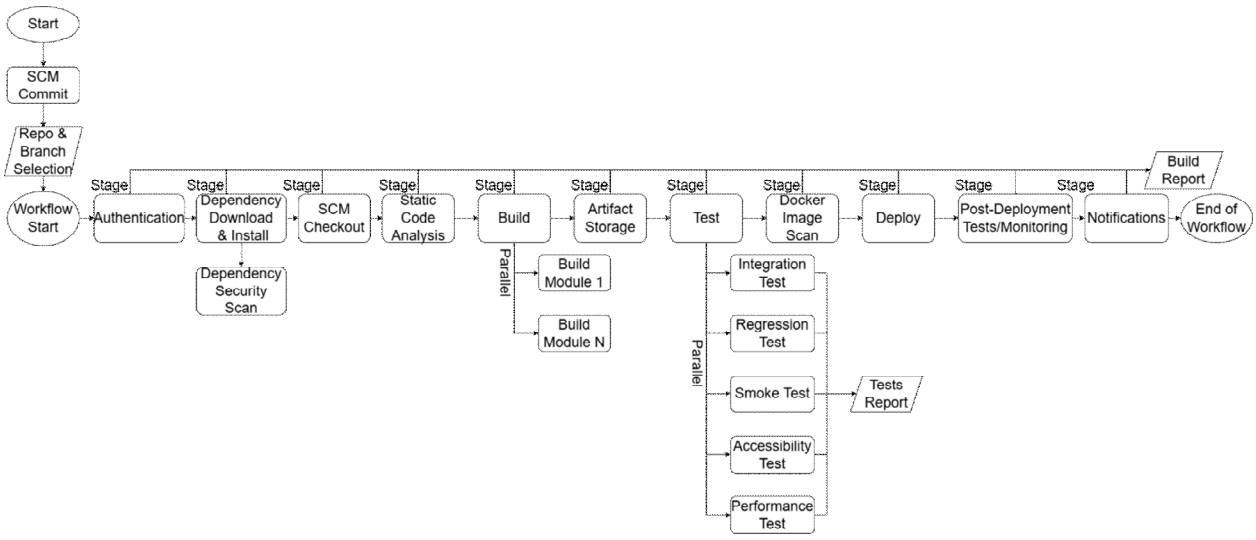

Fig. 2. Architectural scheme of the automated build, test, and deployment Pipeline.
Architectural scheme of the automated build, test, and deployment Pipeline.

5. Conclusions

The conducted research demonstrates that the integration of continuous integration and deployment systems (CI/CD) with modern technologies such as Docker, Kubernetes, ArgoCD, and Amazon Web Services, combined with artificial intelligence algorithms, creates an effective, reproducible, and scalable software development ecosystem. The comprehensive approach allows for the optimization of development processes through early defect prediction and optimization of test scenarios, thereby reducing the time required for bug fixes and enhancing the quality of the final product. Automatic scaling and adaptive resource management, implemented via integrated monitoring systems in Kubernetes and AWS, ensure stable application performance even during peak load periods. Further research aimed at developing more efficient machine learning models and unifying the standards for integrating AI technologies into CI/CD opens additional opportunities for optimizing development processes, which is crucial for ensuring the high efficiency, quality, and adaptability of modern IT systems. The implementation of these approaches in practice could significantly improve software development and deployment processes.

Acknowledgements

The authors express their sincere gratitude to the dissertation research supervisor for valuable recommendations and support throughout the research process. Special thanks are also extended to the staff of the Department of Computer Systems Automation for their assistance and consultations.

Conflict of Interest

The authors declare that there is no conflict of interest.

References

[1] S. Pattanayak, P. Murthy, A. Mehra. Integrating AI into DevOps pipelines: Continuous integration, continuous delivery, and automation in infrastructural management. *International Journal of Science and Research*, Vol. 13(01), 2024, pp. 2244-2256. doi:10.30574/ijsra.2024.13.1.1838

[2] Yara Maha Dolla Ali. Autonomous systems: Challenges and opportunities. *Advances in Engineering Innovation. AEI*, Vol. 4, 2023, pp. 38-42. doi:10.54254/2977-3903/4/2023031

[3] Venkata Mohit Tamanampudi. AI-Augmented Continuous Integration for Dynamic Resource Allocation. *World Journal of Advanced Engineering Technology and Sciences*, Vol. 13(01), 2024, pp. 355-368. doi:10.30574/wjaets.2024.13.1.0425

[4] Osinaka Chukwu Desmond. AI-Powered DevOps: Leveraging machine intelligence for seamless CI/CD and infrastructure optimization. *International Journal of Science and Research Archive*, Vol. 06(02), 2022, pp. 94-107, doi:10.30574/ijsra.2022.6.2.0150

[5] M. Steidl, M. Felderer, R. Ramler. The pipeline for the continuous development of artificial intelligence models—Current state of research and practice. *Journal of Systems and Software*, Vol. 199, 2023, 111615, doi.org/10.1016/j.jss.2023.111615

[6] Yue Zhou, Yue Yu, Bo Ding. Towards MLOps: A case study of ML pipeline platform. *International Conference on Artificial Intelligence and Computer Engineering, ICAICE*, IEEE (2020), pp. 494-500, doi:10.1109/ICAICE51518.2020.00102

[7] Aliyu Enemosah. Enhancing DevOps Efficiency through AI-Driven Predictive Models for Continuous Integration and Deployment Pipelines. *International Journal of Research Publication and Reviews*, Vol. 6(1), 2025, pp. 871-887, doi:10.55248/gengpi.6.0125.0229

[8] Abdul Sajid Mohammed, Venkata Ramana Saddi, Santhosh Kumar Gopal, Dhanasekaran Selvaraj. AI-Driven Continuous Integration and Continuous Deployment in Software Engineering. *2nd International Conference on Disruptive Technologies (ICDT)*, 2024, pp. 531-536, doi:10.1109/ICDT61202.2024.10489475

[9] Kim, G., Humble, J., Debois, P., Willis, J. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. Portland: IT Revolution Press, 2016. – 480 p. URL: https://www.amazon.com/DevOps-Handbook-World-Class-Reliability-Organizations/dp/1942788002

[10] B. S. Satapathy, S. S. Satapathy, S. I. Singh, and J. Chakraborty, "Continuous Integration and Continuous Deployment (CI/CD) Pipeline for the SaaS Documentation Delivery," in *International Conference on Information Technology*, Singapore, 2023, pp. 41-50, doi.org/10.1007/978-981-99-5994-5_5

[11] N. Poulton. *The Kubernetes Book: 2023 Edition*. – JJNP Consulting Limited, 2022. – 3rd edition. – 355 p.

[12] N. Poulton. *Getting Started with Docker: Master the Art of Containerization with Docker*. 2024, Edition. – Packt Publishing Limited – 116 p. doi.org/10.0000/9781835462058-001

[13] Hussain, M. Application of Docker and Kubernetes in large-scale cloud environments. *International Journal of Engineering Research & Technology*, Vol. 12(5), 2023, pp. 450–457, https://doi.org/10.5281/zenodo.8135267

[14] Kubernetes Documentation. Horizontal Pod Autoscaler [Online]. Available: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/ (accessed: 28.03.2025).

[15] Sandeep Pochu, Sai Rama Krishna Nersu, Srikanth Reddy Kathram. Scaling Kubernetes Clusters with AI-Driven Observability for Improved Service Reliability. *Journal of AI-Powered Medical Innovations*, Vol. 3(1), 2024, pp. 39-52, doi.org/10.60087/Japmi.Vol.03.Issue.01.Id.003