International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 08 | Aug - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

Implementing and Using CI/CD: Addressing Key Challenges Faced by Software Developers

Bhaskara Sahithi Shanmukhi

Abstract

Software development teams that want to increase the calibre, dependability, and velocity of their software releases must implement continuous integration and delivery (CI/CD) techniques. Developers must now overcome several obstacles as a result of this shift, including the necessity to automate testing and deployment procedures, uphold strict version control, and promote cooperation between the development, testing, and operations teams. This article looks at the main obstacles that developers face when implementing and using CI/CD, and it does so by analysing case studies and current industry research to shed light on
real-world obstacles and possible solutions.

Keywords

Continuous Integration (CI), Continuous Delivery (CD), Automation, Version Control, Security Challenges, Test Automation, Scalability, Integration Complexity, Tool Compatibility, System Design, Resource Management, Skill Gaps, Workflow Vulnerabilities.

1. INTRODUCTION

Continuous Integration/Continuous Delivery abbreviated as CI/CD is a method that automates integrating, testing, and deploying code changes, ensuring fast and reliable software releases. It supports the larger goal of agile methodology to accelerate the software development lifecycle [1] CI/CD involves two key practices:

1.1 Continuous Integration (CI):

It involves frequent integration and automated testing of the code pushed by developers to a shared repository. This practice ensures that the new code changes do not introduce any bugs guaranteeing the stability and functionality of the codebase.
This process involves aspects like:

- **Automation of Building:** Automation of the process of building the application, making sure that the latest code changes are always in a deployable state.
- **Immediate Feedback:** Developers receive immediate feedback on their code changes, allowing them to act on the issues addressed.
- **Improved Code Quality:** Continuous Integration improves code quality through CI pipelines which often include static code analysis and ensuring the detection of vulnerabilities via security scanning.
- **Performance Monitoring:** To make sure that the performance of the application is not compromised by the most recent code changes, performance monitoring is also applied.

1.2 Continuous Delivery and Continuous Deployment (CD)

- **Continuous Delivery**: The code can now be released into production after passing the integration stage, but it must be done manually.
- **Continuous Deployment:** This takes the concept of CI a step further. In Continuous Deployment the code that passed the automation tests is now ready to be deployed automatically without human intervention. This implies the new code changes and bug fixes will now be delivered quickly and frequently. This ensures that the deployment is reliable and consistent.

CI and CD together form a CI/CD pipeline and boost the software delivery by automating and streamlining the processes of integrating, testing, and deploying code.

1.3 Why CI/CD

- **Increase productivity:** Since everything is automated the time taken to merge, build, test, release, and deploy software is saved here and hence an increment in productivity and efficiency is seen. With this, a developer can focus on writing code of higher quality and monitoring deployments for issues [1].
- **Reliability:** Automation reduces human error, ensuring consistent and reliable processes from code integration to deployment.
- **Quality**: With continuous testing and feedback the code quality is improved leading to the increase of product quality with fewer or no bugs.
- **Lower delivery risk:** Testing each change before deployment guarantees a higher-quality final product and a reduced incidence of defects in production. The development team will spend less time correcting critical flaws found after release, and customers will receive a better product [1].

1.4 A Coin Has Two Sides: The Challenges of CI/CD

While there are numerous impacts of CI/CD, it's important to understand that, much like a coin, the implementation and usage of CI/CD come with their own set of drawbacks.

1.5 Purpose of the Literature Survey

This survey aims to address the challenges faced by software developers while implementing and using CI/CD. It is crucial to recognize these challenges to improve the efficiency, reliability, and overall success of software development. By exploring the existing literature, this survey seeks to highlight the common issues and propose solutions to enhance CI/CD practices.

1.6 Evolution and Pertinence Today

Since its inception, the idea of CI/CD has undergone significant evolution. Software development was first predicated on labour-intensive, prone to error, manual integration, and deployment procedures. A CI/CD solution was developed to automate these procedures in response to the demand for agile methodologies and quicker release cycles. CI/CD is now a fundamental component of contemporary software development, allowing companies to produce high-calibre software rapidly and effectively. As businesses work to meet the demands of rapid development and expansion in a more crowded market, their significance in the sector only grows.

2. Methodology

2.1 Search Strategy

Academic, and professional databases like Google Scholar, ResearchGate, ScienceDirect, and several other online blogs were used to conduct the literature search. Keywords and search terms like “CI/CD challenges”, “Challenges faced by software developers using CI/CD”, "Continuous Integration issues," "Continuous Deployment problems", and "software development CI/CD" were used.

2.2 Inclusion Criteria

The selection criteria for relevant papers included:

1. Papers and blogs published within the last 10 years.
2. Studies that focus on challenges faced by software developers in implementing or using Continuous Integration/Continuous Deployment (CI/CD) pipelines.
3. Research conducted on human software developers or teams.
4. Articles are written in English or with English translations available.
5. Studies with clearly defined methodologies, findings, and implications for CI/CD pipeline usage
6. Articles available in a full-text format for a comprehensive review.

2.3 Selection Process

The titles and abstracts of the papers were used to filter them for relevancy. Next, it was determined whether the full-text articles satisfied the inclusion requirements. For data extraction and citation management ,Zotero was utilized.

3 Thematic Sections

3.1 Integration Issues

3.1.1 Integration Complexity

As companies look to enhance the effectiveness and dependability of their software delivery pipelines, the adoption of continuous integration and continuous deployment (CI/CD) procedures within software development workflows has grown in popularity. However, the integration complexity inherent in these processes creates significant challenges that must be addressed for CI/CD initiatives to be successful. One such challenge is the increased frequency of code changes to the shared codebase, which can lead to conflicts and errors during the integration process. Ensuring that tests are conducted in environments similar to the production environment adds another layer of complexity to the integration process, as it requires replicating various system configurations. Providing developers with seamless access to different versions of the application for testing and deployment is also a complex part of integration. Neglecting integration complexity can lead to greater expenses, slowed-down projects, and lower-quality software.

The impacts of integration complexity include increased costs due to managing code integration and testing environments, project delays caused by the need to resolve integration issues and ensure adequate testing, and

lower-quality software resulting from poor integration techniques that can lead to unnoticed bugs and performance problems. These challenges ultimately affect customer satisfaction and experience.

3.1.2 Tool Compatibility

Tool compatibility in CI/CD refers to the ability of various software tools used within the CI/CD pipeline to work together seamlessly. Compatibility issues can arise from the diverse tool ecosystem, as each tool and technology has its configurations and workflows, making integration complex. It is crucial to ensure that these tools communicate effectively and integrate smoothly without creating conflicts or errors, which ultimately impact the productivity and reliability of the development process. Updates and changes in APIs can break compatibility while tools interact via API, and different versions of the same tool with varying compatibility can lead to integration failures.

When tools aren't compatible, there can be interruptions in the CI/CD pipeline, affecting the entire development process. Debugging and fixing these problems take more time and money, adding to the time and effort required for development. Compatibility problems can also impact the efficiency of the development team by necessitating workarounds and slowing down development cycles, leading to delays.

3.1.3 System Design

In their study, Eero Laukkanen, Juha Itkonen, and Casper Lassenius mentioned that system design issues are common, critical, and understudied. When discussing integration, system design refers to the process of developing and organizing a software system's architecture to ensure that its parts or subsystems can function as a unit. This involves designing the interfaces, data flows, and interaction protocols among various modules, services, and external systems to enable seamless communication, interoperability, and functionality. Effective system design for integration aims to meet the necessary business objectives and ensure that the integrated system functions effectively by addressing issues with compatibility, scalability, data consistency, and performance.

System modularization requires structuring the system into multiple units or services, such as modules or micro services. Each module or service needs to be integrated and tested individually and in conjunction with others, increasing the complexity of managing interactions between modules or services, which can lead to issues with consistency and communication across the system. An unsuitable architecture impacts continuous delivery and integration by making it difficult to deploy changes and test the system effectively. High internal dependencies can complicate the integration process by creating interdependencies that require careful management. Integrating modifications to the database schema with the rest of the system can also be challenging. Updates to data handling procedures and application logic are frequently necessary in response to schema modifications, making it more difficult to maintain data consistency and ensure the seamless operation of the integrated system. Properly managing schema changes is essential to prevent integration problems and maintain system integrity.

3.1.4 Integration Problems

The integration theme covered issues that arise when the source code is integrated into the mainline [4]. Such problems are mentioned below:

- **Large Commits:** Commits with a large number of changes in the code can lead to complications in the integration and increase the risk of conflicts and errors [4].
- **Merge Conflicts:** Conflicts occurring due to unmerged commits, overlapping changes, or when integrating changes from different branches, etc. can impact the integration, which later requires solving them manually [4].

- **Broken Build:** Builds fail due to various reasons like Code Integration Issues, Compilation Errors, Test Failures, and many more can remain broken for an extended period or frequently fail can disrupt the development process and delay progress [4].
- **Work Blockage:** When integration problems or malfunctioning builds because of bottlenecks in the development pipeline, work tasks are impeded or stopped [4].
- **Long-Running Branches:** Branches that stay open for a long time can cause divergent codebases and integration issues, which makes merging more difficult [4].
- **Broken Development Flow:** When problems with continuous integration or other circumstances impede developers' progress and attention, there is a disruption in the development flow [4].
- **Slow Integration Approval**: The efficiency and promptness of integrating new code are impacted by the lengthy approval process for changes to the mainline [4].

3.2 Security Challenges

3.2.1 Security Misconfigurations

Many workflows are configured with overly broad permissions, granting excessive access rights that could be exploited by malicious actors [5].

Improper Permissions: Overly broad permissions in CI/CD workflows can expose the system to risks by granting excessive access rights. This lack of granular access control may allow malicious actors to exploit vulnerabilities, potentially leading to unauthorized changes or data breaches [5]. Implementing the principle of least privilege, where users and processes have only the permissions they need, can mitigate these risks.
Insecure Secret Handling: Insecure handling of secrets, such as API keys and passwords, poses a significant risk in CI/CD environments. When these sensitive credentials are stored or transmitted without proper encryption or access controls, they become susceptible to leaks and unauthorized access. Employing secure storage solutions and rotating secrets regularly are crucial practices to prevent such vulnerabilities [5].
Thus insecure configurations within Continuous Integration (CI) and Continuous Deployment (CD) pipelines can cause the system to have security vulnerabilities or attacks that might compromise the integrity, confidentiality, and availability of the software and CI/CD infrastructure.

3.2.2 Vulnerabilities in Workflows

Talking about the difficulties with workflows on platforms like Github is essential. However, relying on outside actions and parts in CI/CD pipelines adds extra security dangers. These external tools or services might not always adhere to strict security standards, which could lead to weaknesses. Making sure that outside parts are checked for security standards and kept up to date can help reduce these dangers [5]. Getting unauthorized access to repositories and private information can happen because of weak access controls. Hackers can bring weaknesses into the code by submitting pull requests with harmful code. There could be remaining weaknesses and traces if self-hosted runners are used that aren't properly contained or temporary. A self-hosted runner is always a security threat if it gets hacked during a build and keeps harmful traces that impact future builds [6].

3.3 Test Automation Challenges

3.3.1 Flawed Automated Testing

Flawed automated testing, as used in CI/CD, describes the problems that occur when automated tests are not effective or are configured incorrectly. These issues have the potential to damage the CI/CD pipeline's dependability and effectiveness, which could result in missed bugs, sluggish feedback, and subpar products.

Common issues include:

- **Partially Uncovered Tests:** There is a possibility that these automated tests may miss the important application components resulting in unnoticed mistakes or vulnerabilities.
- **Negatives/False Positives:** False positives and false negatives are two ways that tests might give misleading results: they can mistakenly suggest that the code is perfect or fail to find problems that already exist.

3.3.2 Deciding which tests to automate

It is quite a challenging and resource-intensive task to decide which tests should be automated [7] which requires careful consideration of various test types and the management of multiple automation tools. Tests like Unit Tests could be simple to automate but maintaining a large number of unit tests can become burdensome as the codebase grows. Integrate tests can be more complex to set up and maintain than unit tests, especially in environments with multiple interdependent services. Along with the tests tool selection and integration can also be challenging while selecting the right tool from the tests as each tool has diverse strengths and limitations while ensuring seamless integration of various tools within the CI/CD pipeline with tools having different configuration and operational requirements.

3.4 Scalability

An important difficulty in the CI/CD pipeline is scaling up the testing infrastructure as organizations get bigger and manage more data. It takes strategic design and implementation to ensure that the testing system can manage higher data volumes and more complicated infrastructures without sacrificing performance or dependability.
The testing system must process larger datasets to ensure comprehensive testing coverage to handle more data. This increase can lead to longer test execution times and greater demand for storage and processing resources. Infrastructure is becoming more complicated, with more services, dependencies, and components. As a result, the testing system needs to adapt and test these new components efficiently. This may make it difficult to maintain consistent test environments, manage dependencies, and configure systems.

3.5 Skill Gaps

Implementing and maintaining a robust CI/CD pipeline requires a wide range of skills and knowledge, and many organizations face skill gaps that require ongoing training to keep pace with evolving technologies and practices. Essential skills include automation using tools like Jenkins and GitLab CI, familiarity with CI/CD platforms like Travis CI and CircleCI, understanding of DevOps principles, integrating security testing into pipelines, and strong testing skills across unit, integration, end-to-end, and performance testing. Additionally, knowledge of containers (e.g. Docker), instrumentation tools (e.g. Kubernetes), and monitoring and logging tools (e.g. Prometheus and Grafana) are critical. The challenge is to keep up with rapid technological development and ensure the allocation of sufficient resources for capability development and the integration of diverse knowledge to create a unified CI/CD strategy. Addressing these skill gaps through targeted training programs and continuous professional development is essential to optimizing CI/CD processes and successfully delivering software as well as underscoring the importance of observability in maintaining CI/CD pipelines.

3.6 Resource and Cost Management

Implementing CI/CD across the organization has long-term benefits in business agility, product robustness, security, and feature release cycle, but it also has associated costs. The resources, tools, and infrastructure requirements significantly add up to the project costs. Besides, everyone has their preference for tools and other resources as per their convenience. Also, adopting CI/CD doesn't always mean better agility. It can potentially

slow down processes and deter developer productivity if proper guidance and training are not available to developers and executives (Nishant Choudhary, 2024).

Resource limitations are a significant challenge in the implementation and maintenance of CI/CD pipelines. According to the paper 'Problems, Causes, and Solutions When Adopting Continuous Delivery,' one major issue is the limited availability of computational resources. CI/CD pipelines often require substantial CPU and memory to handle the continuous integration and deployment tasks effectively [7] Furthermore, storage constraints pose another challenge, particularly when dealing with large build artefacts, logs, and test data. This limitation can hinder the smooth operation of the pipeline [7]. Additionally, limited network bandwidth can create bottlenecks, affecting the transfer of large files and data within the CI/CD pipeline, and leading to delays and inefficiencies [7]. Addressing these resource limitations is crucial for the successful adoption of CI/CD practices in software development.

4. Conclusion

This literature review explored the diverse landscape of continuous integration and continuous delivery (CI/CD), highlighting both its significant benefits and inherent challenges. The main challenges are

- **Integration complexity:** managing frequent code changes and ensuring a smooth integration process.
- **Tool Compatibility Issues:** Ensuring that different tools used in CI/CD pipelines interact smoothly.
- **System Design Weaknesses:** Addressing architectural and modular issues that hinder CI/CD processes.
- **Security Vulnerability**: Reducing risks related to security flaws and external dependencies. Challenges in test automation: Solving malfunctioning automated tests and selecting appropriate tests for automation.
- **Scalability Issues:** Scale CI/CD processes to handle larger data volumes and complex infrastructures.
- **Capability Gaps:** Addressing gaps in knowledge required to deploy and maintain CI/CD pipelines.
- **Resource Constraints:** Managing the costs and resources associated with CI/CD deployments.

Practical Implications

These challenges highlight the need for organizations to adopt strategic solutions and continuous improvement practices. For example, improving training programs to address skill gaps, investing in compatible tools and robust system design, and implementing strict security protocols can greatly improve CI/CD processes.

Future research directions

- Develop advanced strategies to improve tool compatibility and integration.
- A study of the impact of new technologies such as artificial intelligence and machine learning on CI/CD practices.
- Investigating scalable solutions to manage large-scale CI/CD installations.
- Conducting longitudinal studies to evaluate the long-term effectiveness of various CI/CD practices.

Recommendations

- **Training and Development:** Implement ongoing training programs to equip teams with the necessary skills.
- **Tool Evaluation:** Regularly evaluate and update tools to ensure compatibility and effectiveness.
- **Security measures:** Implement best practices for secure settings and confidential usage.
- **Automated Testing:** Invest in powerful automated testing tools and frameworks for reliability.
- **Scalable Design:** Develop scalable architectures and design resource management to support growing infrastructures.

5. References

[1] Roddewig, S. (2022, February 8). CI/CD: What Is It & Why Is It Important for DevOps? HubSpot. Retrieved from https://blog.hubspot.com/website/cicd

[2] Vemuri, N., Thaneeru, N., & Tatikonda, V. M. (2024). AI-Optimized DevOps for Streamlined Cloud CI/CD. International Journal of Innovative Science and Research Technology, 9(2), 504–505. Retrieved from https://www.researchgate.net/publication/378267974_AIOptimized_DevOps_for_Streamlined_Cloud_CICD

[3] Bhanushali, A. (2023). Challenges and Solutions in Implementing Continuous Integration and Continuous Testing for Agile Quality Assurance. International Journal of Science and Research (IJSR), 12(10). Retrieved from https://www.researchgate.net/publication/375000735_Challenges_and_Solutions_in_Implementing_Continuous_Integration_and_Continuous_Testing_for_Agile_Quality_Assurance

[4] Laukkanen, E., Itkonen, J., & Lassenius, C. (2016). Problems, causes and solutions when adopting continuous delivery—A systematic literature review. Information and Software Technology, 82, 55–79. http://dx.doi.org/10.1016/j.infsof.2016.10.001

[5] Koishybayev, I., Nahapetyan, A., Zachariah, R., Muralee, S., Reaves, B., & Kapravelos, A. (n.d.). Characterizing the Security of GitHub CI Workflows. USENIX. Retrieved from

https://scholar.google.co.in/citations?view_op=view_citation&hl=nl&user=XaUgedIAAAAJ&citation_for_view=XaUgedIAAAAJ:IjCSPb-OGe4C
https://www.usenix.org/system/files/sec22-koishybayev.pdf

[6] Kumar, P., & Madisetti, V. K. (2024). Sher: A Secure Broker for DevSecOps and CI/CD Workflows. Journal of Software Engineering and Applications, 17(5), 321–339. https://doi.org/10.4236/jsea.2024.175018

[7] Coherent Solutions. (n.d.). 4 Common Problems with Continuous Integration and Coherent Solutions. Retrieved from https://www.coherentsolutions.com/insights/continuous-integration-problems