University of the West of Scotland

School of Computing, Engineering and Physical Sciences

**MSc Information Technology**

**MSc Final Project**

**Design and Implementation of Automated CI/CD Pipeline using DevSecOps Practices**

Student name: **Shahid Rasool**

Banner ID: **B01801311**

Email: **shahidchaudhary1119@gmail.com**

Project being undertaken on part-time or full-time basis: **Full Time**

MSc Programme (specify the specialist pathway, if any): **MSc Information Technology (DevOps)**

MSc Programme Leader:

**Contents**

**Abstract**

**1. Introduction to DevSecOps**

**2. Research and Literature Review**

**2.1 Introduction**

**2.2 Continuous Integration and Continuous Deployment (CI/CD)**

**2.3 DevOps: Principles and Evolution**

**2.4 DevSecOps: Integrating Security into CI/CD Pipelines**

**2.5 Best Practices and Tool Integration**

**2.6 Summary and Research Gap**

**6. References**

**Abstract**

Security is often treated as an afterthought in the software development life cycle, even though a single breach can compromise the stability and reliability of an entire system. To achieve secure and continuous software delivery, security practices must be embedded throughout all stages of development rather than appended as a final step. In modern development environments, where developers frequently push code to central repositories multiple times a day, manual build and deployment processes are time-consuming, error-prone, and reliant on human oversight. These challenges underscore the necessity for automated systems that can manage builds, testing, and deployments efficiently while incorporating security controls at every stage of the pipeline.

This project proposes the design and implementation of an automated Continuous Integration and Continuous Deployment (CI/CD) pipeline integrated with DevSecOps practices. The proposed solution aims to streamline the software delivery process, reduce human error, and ensure that essential security checks are conducted automatically. The research is motivated by the growing demand for secure automation in software engineering and the limited availability of practical academic models demonstrating the integration of DevSecOps within CI/CD workflows.

The primary aim of this project is to design, implement, and evaluate an automated CI/CD pipeline with integrated DevSecOps mechanisms that enhance both the efficiency and security of software development and deployment processes.

1. **Introduction to DevSecOps**

DevSecOps, which merges Development, Security, and Operations, is a modern software engineering methodology that embeds security practices into every phase of the DevOps workflow. It emphasizes collaboration among developers, operations engineers, and security professionals to ensure that software is developed securely, vulnerabilities are detected early, and risks are minimized throughout the software development lifecycle. The key philosophy behind DevSecOps is the *“shift-left”* principle — integrating security measures during the earliest stages of development rather than treating them as post-deployment activities (Tigera, 2025). By automating security checks and promoting a culture of shared responsibility, DevSecOps enhances an organization’s overall security posture while accelerating time to market and improving resilience against evolving threats.

As DevOps continues to gain widespread adoption, the importance of integrating security within its continuous integration and delivery processes has become increasingly evident. However, according to a global survey conducted by HPE Security Fortify (2016), although most professionals acknowledge that security should form an integral part of DevOps, implementation remains limited across enterprises. Supporting this finding, Gartner reported that fewer than 20% of enterprise security architects actively engage in DevOps practices to embed information security into these workflows. This disparity between awareness and implementation underscores the emergence of **DevSecOps** — a framework specifically designed to integrate security as a core component of the DevOps lifecycle (Mora M. O., 2017).

Within the context of this research, DevSecOps is examined as a mechanism to improve both **software delivery speed and security** by embedding automated security tools and governance mechanisms directly into the CI/CD pipeline. This integration aligns with the study’s aim to evaluate how automated DevSecOps pipelines can balance efficiency and protection in modern software delivery environments.

1. **Research and Literature Review**
   1. **. Introduction**

Modern software engineering increasingly relies on automation to achieve faster and more reliable software delivery. Continuous Integration and Continuous Deployment (CI/CD) pipelines have become central to this transformation, enabling teams to automate code integration, testing, and deployment processes. However, while CI/CD practices improve delivery speed, they often lack embedded security mechanisms (Rajapakse & Nandigam, 2022). This limitation has given rise to the concept of DevSecOps, which extends DevOps by integrating security at every stage of the development lifecycle. This section critically reviews existing research on CI/CD, DevOps, and DevSecOps principles, identifying best practices, tools, and gaps in the current body of knowledge relevant to this project.

**2.2 Continuous Integration and Continuous Deployment (CI/CD)**

**Continuous Integration (CI)** is the practice of frequently merging code changes into a shared repository where automated builds and tests are executed. It enables early detection of integration issues and supports consistent software quality (Fowler, 2006). **Continuous Deployment (CD)** extends CI by automatically deploying validated code to production, reducing manual intervention and delivery time (Bass, Weber & Zhu, 2015). Together, CI/CD forms the foundation of modern software automation pipelines, offering benefits such as faster release cycles, reduced human error, and improved feedback loops (Atlassian, 2023).

Recent industry studies highlight CI/CD adoption as a key factor in achieving high software delivery performance. According to the (Puppet Labs, 2022) *State of DevOps Report* (2022), organizations that fully automate their delivery pipelines achieve up to 200 times faster deployment frequencies and 24 times faster recovery from failures compared to low-performing teams. However, such automation can inadvertently accelerate the delivery of insecure code if security is not integrated throughout the pipeline. This realization has driven research towards combining CI/CD with proactive security practices — leading directly to the DevSecOps paradigm.

**2.3 DevOps: Principles and Evolution**

**DevOps** emerged as a cultural and technical movement that fosters collaboration between development and operations teams to deliver software more efficiently and reliably (Kim et al., 2016). It emphasizes automation, continuous testing, and monitoring, resulting in improved agility and reduced time-to-market. DevOps bridges the gap between development and operations through shared responsibility, iterative workflows, and the use of infrastructure automation tools such as Docker, Kubernetes, and Terraform (Bass et al., 2015).

However, despite its success, DevOps initially treated security as an external or late-stage concern. According to Gartner (2016), fewer than 20% of enterprise security teams actively participate in DevOps processes, leading to unaddressed vulnerabilities and compliance risks. This gap catalyzed the evolution toward **DevSecOps**, a framework designed to embed security and compliance within the DevOps lifecycle itself.

**2.4 DevSecOps: Integrating Security into CI/CD Pipelines**

**DevSecOps** integrates *security controls, processes, and culture* into DevOps pipelines. Its central principle, known as *“shift-left security,”* emphasizes embedding security practices early in the development lifecycle rather than applying them as a final gate (Rajapakse & Nandigam, 2022). By automating vulnerability scanning, static code analysis, dependency management, and container security, DevSecOps ensures that each code commit undergoes rigorous security validation without delaying delivery (Tigera, 2025).

In practice, tools such as **SonarCloud**, **OWASP Dependency-Check**, and **Trivy** are integrated into CI pipelines to detect security flaws at different layers — source code, dependencies, and container images respectively (Security, 2023). **Infrastructure as Code (IaC)** tools like **Terraform** further support security automation by enforcing policies and defining infrastructure configurations in code, ensuring consistent and auditable environments (HashiCorp, 2023)A. The combination of CI/CD automation and DevSecOps practices reduces both deployment time and post-release vulnerabilities, enhancing overall system resilience.

Nevertheless, empirical research on quantifying these benefits remains limited. Most existing literature focuses on conceptual frameworks or qualitative assessments of DevSecOps adoption. Few studies provide measurable evidence on how integrating security automation influences key performance indicators such as build time, deployment frequency, or vulnerability reduction this project directly addresses.

**2.5 Best Practices and Tool Integration**

The successful implementation of DevSecOps within CI/CD pipelines depends on aligning tools and workflows that support automation, transparency, and collaboration. GitHub Actions and Jenkins remain leading CI/CD tools, with GitHub Actions providing a cloud-native, event-driven workflow ideal for smaller or cloud-hosted projects, while Jenkins offers extensive customization for enterprise environments (Atlassian, 2023). For IaC, Terraform outperforms AWSCloudFormation in terms of multi-cloud portability and modularity, though CloudFormation remains advantageous for AWS-only deployments (HashiCorp, 2023).

A comparison of common tools used in DevSecOps pipelines is summarized below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Purpose** | **Tools** | **Description** | **Advantage** |
| CI/CD | Github Actions | Cloud-native automation workflows | Simpler setup, GitHub integration |
| IaC | Terraform | Declarative infrastructure provisioning | Multi-cloud, reusable modules |
| Static Code Analysis | SonarCloud | Cloud-based SAST and code quality | Easy integration with Cl |
| Dependency Scanning | OWASP Dependency-Check | |  | | --- | | Detects vulnerable libraries |  |  | | --- | |  | | |  | | --- | | Open-source, standards-based |  |  | | --- | |  | |
| |  | | --- | | Container Scanning |  |  | | --- | |  | | |  | | --- | | Trivy |  |  | | --- | |  | | |  | | --- | | Scans Docker images for CVEs |  |  | | --- | |  | | Fast and developer-friendly |

These tools, when orchestrated in an automated pipeline, enable continuous validation of both functionality and security. Integrating them in a single automated workflow reflects current industry best practice and aligns with DevSecOps maturity models promoted by AWS and GitLab (AWS, 2023; GitLab, 2024).

**2.6 Summary and Research Gap**

Existing literature and industry research confirm that integrating security into DevOps improves software quality, compliance, and response to threats. However, there remains a lack of empirical, academic evidence demonstrating how DevSecOps integration affects both delivery speed and security outcomes in measurable terms. Most frameworks describe conceptual benefits rather than presenting quantitative data.

This project contributes to that gap by designing and implementing a fully automated CI/CD pipeline with embedded DevSecOps tools (SonarCloud, OWASP Dependency-Check, Trivy, Terraform) deployed on AWS. By collecting and analysing metrics such as build time, deployment frequency, and vulnerability counts, this study aims to evaluate the real-world trade-off between security and speed — directly addressing the stated research question.

This synthesis of literature demonstrates the theoretical foundation for integrating security automation within CI/CD pipelines. The next stage of this research will operationalise these findings by designing and implementing an automated DevSecOps pipeline on AWS, allowing quantitative measurement of build time, deployment frequency, and vulnerability reduction.

**3. References**

Atlassian. (2023). *Continuous Integration vs Continuous Deployment*. Retrieved from Atlassian.

Bass, L., Weber, I., & Zhu, L. (2015). *DevOps: A Software Architect’s Perspective.* Boston, MA: Addison-Wesley.

Fowler, M. (2006). *Continuous Integration*. Retrieved from Martin Fowler: https://martinfowler.com/articles/continuousIntegration.html

Gartner Research. (2016). *Integrating Security into DevOps: The Future of SecOps.* Stamford, CT: Gartner Inc.

HashiCorp. (2023). *Terraform Documenation*. Retrieved from HashiCorp Developer: https://developer.hashicorp.com/terraform

Hewlett Packard Enterprise. (2016). *Application Security and DevOps Survey.* Palo Alto, CA: HPE Security Fortify.

Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The DevOps Handbook.* Portland, OR: IT Revolution Press.

Mora, M. O. (2017). An Analysis of DevSecOps Adoption in Software Development Practice. *Journal of Software: Evolution and Process*, 45-60.

Mora, M., O’Connor, R. V., Rout, T. P., & Dorling, A. (2017). Software Process Improvement and Capability Determination: 17th International Conference, SPICE 2017, Palma de Mallorca, Spain, October 4–5, 2017, Proceedings. *SPICE 2017.* Cham: Springer.

Puppet Labs. (2022). *State of DevOps Report.* Portland, OR: Puppet.

Rajapakse, J., & Nandigam, J. (2022). DevSecOps: Integrating Security into DevOps Pipelines. *Journal of Software Engineering and Applications*, 145-162.

Security, A. (2023). *Trivy Open Source Vulnerability Scanner*. Retrieved from Aqua Security.

Services, A. W. (2023, October 28). *Introduction to DevSecOps on AWS*. Retrieved from Amazon Web Services Whitepaper: https://aws.amazon.com/what-is/devsecops/

Tigera. (2025, October 28). *What Is DevSecOps and What Do You Need to Succeed?* Retrieved from Tigera.io: https://www.tigera.io/learn/guides/devsecops/