JCBI
Journal of Computing & Biomedical Informatics
ISSN: 2710 - 1606
Volume 08 Issue 02
2025

Research Article
https://doi.org/10.56979/802/2025

Software Development Empowered and Secured by Integrating A DevSecOps Design

Samavia Riaz[1], Ayyan Asif[2], Younus Khan[3], Muhammad Ibrar[3], Saira Afzal[4], Khalid Hamid[5*], Sehar Gul[6], and Muhammad Waseem Iqbal[7]

[1]Computer Engineering, Department of Computer Systems Engineering, The Islamia University of Bahawalpur, Paksitan.
[2]Master of Science in Data Analytics (Stem), Department of Computer Science, New Mexico State University, Las Cruces, NM, USA.
[3]Department of Computer and Mathematical Sciences New Mexico Highlands University, Las Vegas, NM; USA.
[4]Department of Computer Science, The Sahara College Narowal, Pakistan.
[5]Department of Computer Science, Superior University Lahore, 54000, Pakistan.
[6]Department of Computer Science, Sukkur IBA University, Sindh, Pakistan.
[7]Department of Software Engineering, Superior University Lahore, 54000, Pakistan.
*Corresponding Author: Khalid Hamid. Email:khalid6140@gmail.com

Received: November 11, 2024 **Accepted:** March 01, 2025

Abstract: This made the development of software grow fast, injecting speed and agility in the processes of delivery of software, but integrating security into these high-speed environments has remained a challenge. The solution to this problem comes through the adoption of a methodology known as DevSecOps, encompassing security at each step in the lifecycle of software development. It explored the adoption and value of DevSecOps, concentrating more on automation, vulnerability detection, and continuous security testing. It outlines a comprehensive review of available literature on the topic, with a special focus on the leading tools in this list, namely Static Application Security Testing (SAST), Software Composition Analysis (SCA), and Dynamic Application Security Testing (DAST). The paper will go on to discuss real examples of DevSecOps implementation and follow that up with a discussion of emerging trends, such as machine learning, cloud-native security, and zero-trust models. The study depicts the fact that, though DevSecOps has not matured as a concept yet, its adoption is at a very critical phase in building secure, efficient, and resilient software systems.

Keywords: DevSecOps; Software Development; Static Application Security Testing; DevOps; Life Cycle; Agility.

1. Introduction

DevSecOps is a concept that has become popular today in the fast-paced world of software development and streamlines a process much better to work at the speed required and to achieve efficiency faster. But challenges have their place in the concerned issue at the same speed. Security is the element where DevSecOps proves itself different: It integrates security into every phase of the software development process [1], [2].

This practice is between developers, operations teams, and security teams integrating security early and continuously into the development pipeline [3]. DevSecOps utilizes automation tools to better discover and remediate vulnerabilities with a higher speed-to-value whilst coming at a minimal cost [4]. Examples include SAST (Static Application Security Testing), and DAST (Dynamic Application Security Testing) [5]. The concept here is to look for problems earlier in the development lifecycle before they progress into advanced stages of development, hence costly to cure and time-consuming [6].

This paper will discuss the current status of DevSecOps, its advantages, as well as some problems it meets. For this purpose, we will take into consideration the practices of leading companies that successfully applied DevSecOps for production purposes and the latest trend within the framework:

machine learning, cloud-native security, and zero-trust models, changing its future [7], [8]. Though DevSecOps is still in the very early period of evolution, it is quite obvious that it provides a more efficient, secure, and resilient way of developing software [9], [10].

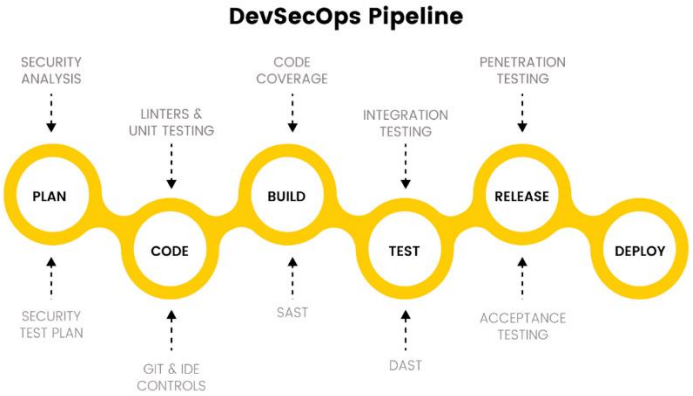

Figure 1. DevSecOps pipeline [31].

2. Literature Review

In DevSecOps, security should therefore be integrated into every stage of the software development process rather than being bolted at the tail end. In this way, companies can develop software that is not only more secure and cyber-proof. The paper looks at how DevSecOps can improve security by automating security tasks and testing for resilience while embracing all the new trends. Security checks and testing will be performed early so that vulnerabilities will be caught sooner, and the time and cost to have errors corrected can be reduced. It also presents real cases of companies using DevSecOps to enhance their security and performance at work. Trends such as machine learning, cloud-native security, and zero-trust models will help determine what this will look like in the future for DevSecOps, so organizations are better positioned to build software that is even stronger and safer [11].

DevSecOps is all about making sure security is built into every part of the software development process. The paper explores how DevSecOps can improve security, focusing on automating security tasks, and testing DevOps involves the delivery of software quickly by speeding up the development and deployment process; however, these imply risks associated with security issues. It examines this concept of DevSecOps in detail concerning security, its current state in the industry, and what challenges are being experienced when companies are trying to implement this. It appears that there is a review of grey literature and industry surveys in the paper relating to China, which identifies such factors making the practice insecure as: "security is optional when speed matters." DevSecOps is categorized into three key aspects such as capabilities, cultural enablers, and technological enablers. Best practices on people, processes, and technology are suggested that can be helpful for companies as they start integrating security into their workflows. It is still in the budding stage, and this idea has begun to draw attention as the world watches and waits for increased research potential in academic and industrial environments. For resilience, and embracing new trends. The paper also highlights real-world examples of companies successfully using DevSecOps. Looking ahead, trends like machine learning, cloud-native security, and zero-trust models are set to shape the future of DevSecOps, helping organizations build even stronger, safer software [12].

DevSecOps adds some high-level security measures to the process of DevOps and learns how to secure applications and data without losing speed or flexibility [13]. It continuously updates security techniques and uses open-source tools that show how security can be embedded in a development pipeline. This approach may take more resources and time to integrate, but it responds to possible threats quicker than the rest, thereby preventing attacks like data breaches. DevSecOps might have the ability to assist developers detect vulnerabilities as soon as the development process starts so that the code remains secure, thus preserving efficiency for DevOps-driven workflows by running regular security checks[14].

The DevSecOps pipeline with SAST, SCA, and DAST became pretty efficient in terms of the discovery of vulnerabilities at different stages in the process of software development. Here, SAST was mainly concerned about finding weaknesses in code, while SCA was used for highlighting issues about

third-party components, and DAST picked on runtime vulnerabilities, later in the process. Altogether, these were going to provide the DevSecOps pipeline with substantial security coverage. Collaborations with the teams in development, security, and operations provide an early look at the vulnerabilities. The study elucidates how the integration of SAST and SCA into the CI/CD process significantly decreases the introduction of security issues, while DAST enhances runtime security. The research offers a blueprint for implementing these security techniques in a DevSecOps pipeline to ensure the proactive securing of software and shared responsibility for security culture within the organizations[15].

This paper identifies and ranks the challenges concerning the adoption of DevSecOps: security integrated into the DevOps process, ensuring secure software. A review of the literature and a survey found 18 key challenges that are categorized under 10 main categories. The paper shows that "standards" have the most influential role in affecting other challenges. The study used a methodology named fuzzy TOPSIS and identified three major issues: secure coding standards, automated security testing tools in DevOps, and knowledge about static security testing. It underlines the need for solving these issues for adopting DevSecOps among organizations with effectively developing software in a secure way [16].

This paper gives an overview of the security challenges that are associated with cloud-based software development and promotes the incorporation of security into all stages of development through DevSecOps. While cloud computing has transformed software development, it also brought problems such as data breaches, misconfigurations, and insecure interfaces into this domain. To alleviate these problems. The research emphasizes secure coding, threat modeling, and regulatory compliance as the most important aspects of building a secure, cloud-native application. Best practices for implementation in the context of DevSecOps in the cloud are presented to help organizations bolster security without sacrificing the benefits of agile, scalable cloud computing. The paper concludes by prescribing security training, involving champions of security, and cultivating a culture with an awareness of security to improve cloud security [17].

The paper has synthesized the existing literature and highlighted the challenges that emerge in the implementation of the DevSecOps model-in which security is infused in the DevOps process for securely delivering software without compromising on agility through the use of 52 peer-reviewed studies. There were identified 21 specific challenges to implementing DevSecOps and 31 proposed solutions. Most of the challenges emerged from the tools, thus tending toward requiring automation in security practices. Human factors are extremely critical for successful adoption but have been less studied. This work aims to help practice with anticipation of its difficulties and explore the solutions that can then be built upon for future research in this area [18].

DevSecOps is a research approach in which security practices are integrated into the DevOps methodology to make software development stronger while protecting customer data and the rule of law. In the growing role of security in fastening development cycles, this research aims to understand key metrics that organizations can use to measure the effectiveness of their implementation of DevSecOps. To realize this, the authors conducted a Multivocal Literature Review (MLR), examining different grey literature sources to gather insights from both professionals and academics. According to the study, establishing effective metrics is extremely crucial in the improvement of measurement practices in software engineering and information systems. The paper takes a structure of the theory of DevOps and DevSecOps, elaborates the methodology of research, finds out results, and discusses the implications, and problems or limitations of the study. Finally, it would contribute to improving an understanding of the concept of DevSecOps and measurement in an organization [19].

This article is majorly on the importance of vulnerability management within a DevSecOps framework: integration of security practice in every phase of software development. Vulnerability management is part of DevSecOps that involves identifying weak points in software systems, assessing the same for improvement, and helping the system fix them to, as such, secure and prevent security breaches. Two good tools used for vulnerability detection are SAST and DAST. IAM and configuration management are also very important to secure systems. Periodic evaluations and updating vulnerability management programs ensure that the organization's systems and other functions are always ahead of threats but constantly secure and compliant with industry standards[20].

This thesis focuses on the ever-increasing threat of securing in a fast world of DevOps, primarily because most organizations deploy software fast through automated pipelines. When DevOps accelerates

the release of software, consequences bring risks, such as security when securing deployment pipelines; and securing deployment pipelines amidst balancing speed and security with insider threats. The research study will employ a Systematic Literature Review of 18 articles in its quest to integrate security into DevOps in the pursuit of developing it into DevSecOps. The results suggest that as of now, the research is more targeted toward the acquisition of infrastructure such as containers and cloud environments while completely neglecting the cultural factors such as the inculcation of a security mindset among teams. In short, putting security into DevOps, or DevSecOps, involves a change in culture, automation, and collaboration across the development, operations, and security teams [21].

This paper discusses how DevSecOps can be integrated into the CI/CD pipeline to improve security while providing fast software releases. DevSecOps is an enhancement of DevOps as it adds security to the development and operations processes. In light of increasing cyber-attacks, organizations using CI/CD have to adopt strong security practices speedily. This paper indicates some critical challenges identified in the paper, such as the lack of security skills in many teams using DevOps, and proposes solutions such as automation and security testing tools such as SAST and DAST. It particularly highlights the fact that security has to be integrated into every step of the pipeline, emphasizing access control, managing secrets, and continuous testing of code to reduce the risk of insecure code. In a nutshell, securing the CI/CD pipeline is necessary for developing better and safer software [22].

This paper explores the integration of security into fast-paced DevOps and agile development environments [23]. It aims to formulate a Secure Software Development Model that would identify security flaws easily at the earliest phases of development, not taking much time and adding less cost to the model [24], [25], [26]. The study focuses on one Finnish software company named Awake.AI in the maritime industry and looks into its security practices. Topics of interest include cloud computing, Docker, Kubernetes, and DevSecOps, where recommendations regarding security will be presented. It includes the ability of automated scanners that find vulnerabilities at the initial stages with the least possible risks. For this reason, the approach followed by Awake.AI was ideal since it enhanced security while the development process was not affected. The future works can be in the enhancements of documentation practices and proper balance in the addition of new features and preventing vulnerabilities. A strong security framework will ensure that customer trust is assured and value is added to the organization [27].

3. Methodology

This research uses a secondary data-based approach to investigate the incorporation of security into the DevOps lifecycle. The methodology adopted includes an in-depth review and synthesis of the existing literature available in academic articles, industry reports, case studies, and whitepapers to provide a holistic overview of the DevSecOps frameworks, tools, and implementation strategies [28], [29], [30].

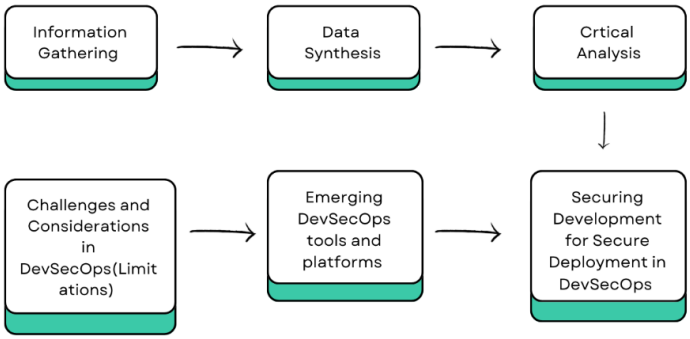

Figure 2. Study methodology

3.1. Information Gathering

Use academic databases and search engines to gather pertinent literature. Search results from pertinent literature using terms such as "DevSecOps", "secure DevOps," "DevOps security", and "software security", "as well as their variants in research, academic studies, news, reports, and journals in which peer-reviewed articles took precedence in all instances because it reflects relevant, accurate information that was within the stipulated timeline of recency for data considered credible.

3.2. Evaluation of Literature

The selected articles and reports were critically examined to determine: the core principles and practices of DevSecOps; key integration challenges of security into the software development lifecycle; tools and frameworks; and case studies involving real-world implementation of DevSecOps.

Each piece of literature was reviewed concerning its contribution toward the field, reliability, and potential to enhance understanding of the DevSecOps model. Articles that provide empirical data, innovative approaches, or detailed insights into challenges and their solutions were prioritized.

3.3. Data Synthesis

The collected data was systematically synthesized to:

1. Identify Patterns and Trends: Common themes, challenges, and best practices in DevSecOps were extracted and summarized.
2. Highlight Best Practices: Practical recommendations for implementing DevSecOps in organizational settings were outlined.
3. Examine Case Studies: Case studies were analyzed to understand the adoption, effectiveness, and outcomes of DevSecOps in various industries.

This process included comparing findings across different sources to ensure consistency and uncover gaps in current research.

3.4. Critical Analysis

A critical analysis of synthesized data was conducted for:

- Drawing focus on the current status quo of DevSecOps adoption.
- Areas for improvement and potential areas of future research Identify knowledge gaps, such as scalability, automation, and risk-based security strategy gaps.

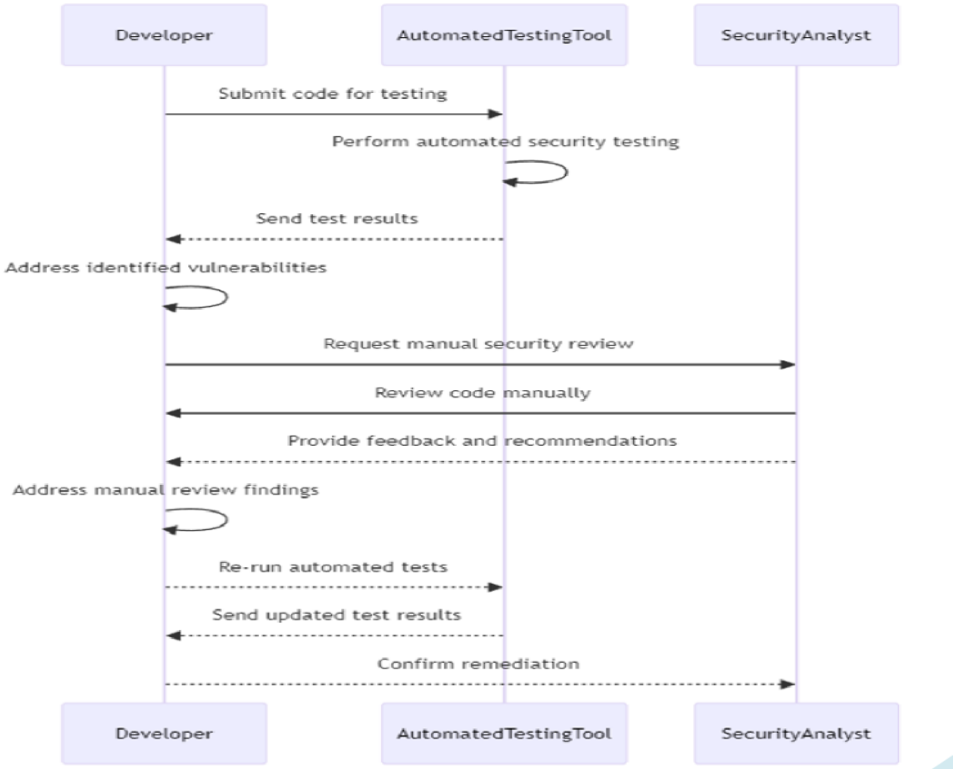

Figure 3. Critical analysis flow [31]

3.5. Securing Development for Secure Deployment in DevSecOps

In a DevSecOps framework, securing the development process is about putting security practices and controls at every stage of the software development lifecycle. That way, applications are designed to resist cyberattacks from the outset. The following are key strategies and considerations for preparing code for secure deployment.

1. Code Review and Security Testing

Code Reviews: Developers review each other's code for flaws and ensure security best practices are followed. Such peer reviews catch vulnerabilities early.

Automated Security Testing Tools can automatically scan your code for such common security problems as:

Injection Vulnerability

- Cross-Site Scripting (XSS)
- Unsafe Configuration
- Vulnerability discovery and elimination would reduce the risk of security incidents and save remediation costs.

2. Secure Coding Practices

Training should include secure coding basics, which include: Input validation to prevent unauthorized entry of data. Output encoding against attacks. Correct error handling that doesn't leak information. Secure frameworks: Libraries and components that have been validated before inclusion into the system are less risky as they would follow the industrial standard. This in turn lowers the possibility of the vulnerabilities coming into the code base due to the culture of secure coding.

3. Static Application Security Testing (SAST)

SAST tools examine source code or binaries for flaws that may lead to security breaches. By integrating SAST into the Continuous Integration (CI) pipeline:

- Developers can automatically scan for security risks.
- Issues are resolved early in the development process.

4. Dependency Management

The modern application relies significantly on third-party libraries and frameworks. So, to secure them, one must:

Maintain and periodically update all dependencies, ensuring patched versions for known vulnerabilities.

-Use tools such as Software Composition Analysis (SCA) and dependency scanners that are used to identify risks in third-party components.

-It's a proactive act reducing the chances of security threats based on third-party code.

5. Secure Configuration Management

Ensure secure configurations of development environments: Tools, libraries, and frameworks must be configured following the best practices. Tools and configurations must be automated by employing automated scripts and configuration management systems to enforce the stringent application of secure settings. Misconfigurations are significantly reduced by the proper implementation of configuration management, thus strengthening the security of the development environment.

Table 1. Overview of emerging DevSecOps tools and platforms

Tools/ Platform	Functionality	Adoption Rate	Key Feature	Benefits & Optimized Understanding	Security Concerns
GitLab	IntegratedDevOps platform with built-in security features, including static and dynamic security testing, container canning, and vulnerability management.	High	Continuous integration and continuous delivery (CI/CD) pipeline Built-in security testing and compliance checks.	Unified platform for DevOps and security; close integration with version control processes, and CI/CD.	Potential over-reliance on the built-in tools might miss other security solutions; and complexity in configuration for some advanced security settings.

Jenkins	An open-source automation server supports Descopes practices through plugin integration for security testing, code analysis, and deployment automation.	Moderate	Extensive plugin ecosystem for customizing CI/CD workflows Integration with security testing tools such as OWASP Dependency-Check and SonarQube	Highly customizable with the large ecosystem of plugins; integrates quite well with numerous security testing tools.	Complex plugin management; and configuration issues can allow for vulnerabilities to exist.
CircleCI	infrastructures as Code (IaC) Tool enables developers to provision and manage cloud infrastructure resources using declarative configuration files.	Moderate	Automated workflows for building, testing, and deploying applications Integration with GitHub, Bitbucket, and GitLab for version control	Automating the workflow simplifies infrastructure management, making deployment speed more agile.	Potentially vulnerable to misconfigured infrastructure code; potentially risky third-party integrations, when not adequately vetted.
Docker	Cloud-based CI/CD platform offering native integrations with security testing tools, version control systems, and cloud platforms	High	Containerization for consistent deployment environments Security features like Docker Content Trust for image signing	Ensures consistent environments from development to production; ideal for microservices and cloud-native applications.	Container image vulnerabilities; must ensure images are signed and scanned for security issues.

k8s	The containerization platform enables developers to package applications and dependencies into lightweight, portable containers.	High	Automated deployment and scaling of containerized applications Integration with security tools for vulnerability scanning and runtime protection	Scalable and resilient for microservices; automation reduces manual intervention and improves deployment speed.	This could expose the system to threats in case of misconfiguration in access control or cluster security; API and container runtime need to be secured.
AWS Code Pipeline	A container orchestration platform automates, scales, and manages containerized applications.	High	Integration with AWS services like AWS Code Commit, Code Build, and Code Deploy Security features like encryption, access control, and compliance audits.	It natively integrates with the AWS ecosystem, and supports encryption and compliance checks.	Dependence on AWS-specific security tools; requires expertise in managing AWS security configurations.

3.6. Objective

This study aims to synthesize secondary data to arrive at actionable insights into improving the resilience of software security through the seamless integration of security into the DevOps lifecycle. It provides a structured way of ensuring the findings are complete, relevant, and applicable to let organizations embrace DevSecOps principles effectively while trying to address the evolving challenges of software security.

3.7. Proposed Models for Securing the DevOps Lifecycle

To implement effective DevSecOps, security should be integrated into all stages of the development lifecycle. The following models outline some of the best practical approaches toward secure coding, automated vulnerability detection, and secure deployments.

3.8. Comprehensive CI/CD Pipeline with Integrated Security (Diagram Below)

This design portrays an end-to-end secure CI/CD process while focusing on necessary steps involving the integration of security with the entire process.

1. Codebase and Dependency Checks

Codebase and dependency verification: The programmers sit in front of an integrated development environment writing codes. Even before the commit into the VCS, dependencies verify whether it makes use of outdated libraries, versions, etc. It implements the stage for analyzing the code's quality which allows catching a critical coding issue.

2. Secure Docker Image Creation

The pipeline constructs Docker images for containerized deployments and scans the Docker images for vulnerabilities to ensure that no vulnerability exists in the images.

3. CI/CD Workflow Automation

GitHub Actions handle the CI/CD pipeline; they automate workflows like pushing images to Docker Hub after security scans.

4. Monitoring and Alerts

After deployment, Kubernetes monitoring tools such as Prometheus and Grafana track the health of the application and detect potential threats in real time.

5. Provisioning Secure Infrastructure

The whole application is deployed on a secure EKS (Elastic Kubernetes Service) cluster in a scalable and protected form.

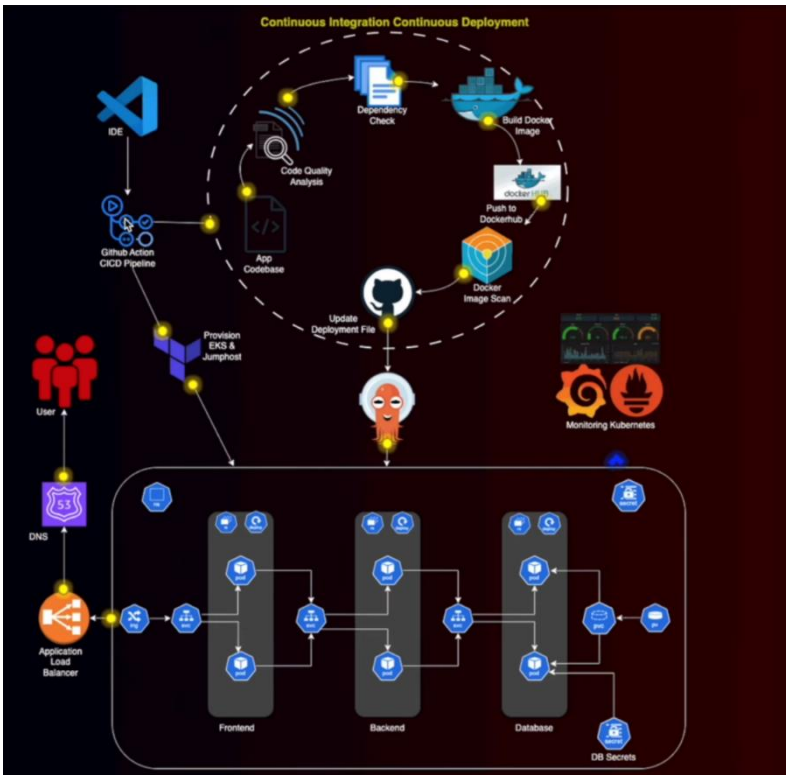

Figure 4. DevSecOps proposed model [32]

3.9. Security-Oriented DevOps Workflow Using ArgoCD (Diagram Following)

This is a pattern of using tools, such as Terraform, Jenkins, and ArgoCD, that assimilate security into provisioning infrastructure and deploying applications.

1. Infrastructure as Code (IaC)

With Terraform, the infrastructure is secured by defining and deploying the infrastructure. This guarantees consistent and auditable infrastructure configuration.

2. CI/CD Pipeline Integration

A Jenkins-based CI/CD pipeline provides a robust CI/CD pipeline that makes:

i. File scans for malicious files.

ii. Docker image building and scanning using a tool like Trivy.

iii. Images are securely pushed into the AWS Elastic Container Registry (ECR).

3. GitOps for Deployment

ArgoCD enables the practice of GitOps to provide traceability and versioning for secure deployment.

4. Enhanced Monitoring and Alerts

Logging and monitoring tools like ELK, Prometheus, and Grafana are used to give deeper insights into the performance and security of the application.

5. Secure Application Layer

The application consists of multiple microservices, such as Web App, API Gateway, Position Tracker, etc., deployed securely using Kubernetes.

Backend databases are protected with encrypted storage mechanisms like EBS (Elastic Block Store).

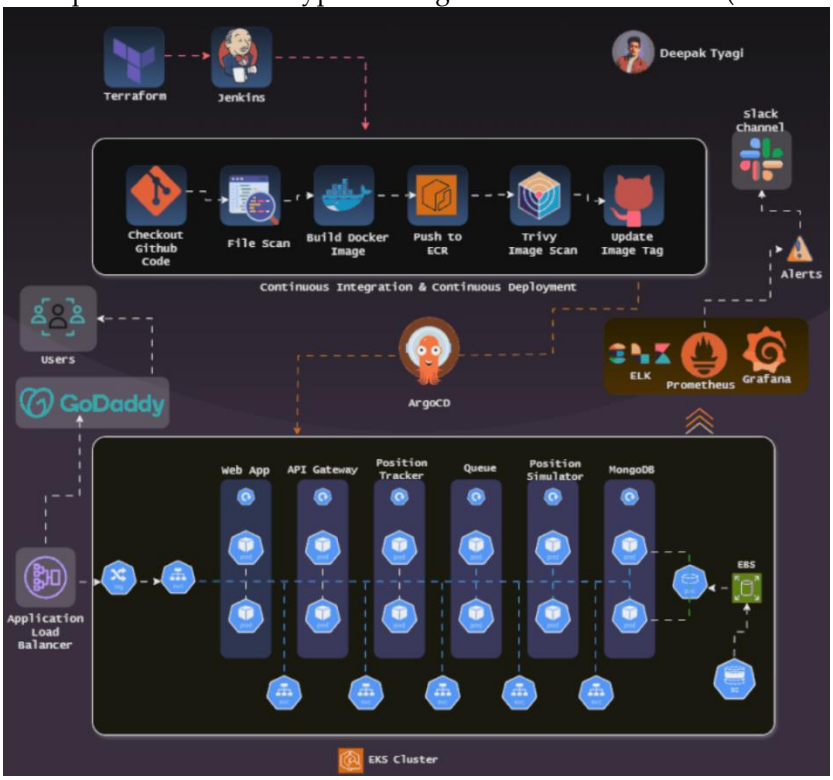

Figure 5. Security-Oriented DevOps workflow using ArgoCD [33]

3.10. Challenges and Considerations in DevSecOps (Limitations)

Despite being highly secure and efficient, DevSecOps has a few challenges

1. Resistance to Change

Classic roles and silos are often unwilling to accept the collaborative culture that DevSecOps requires. Strong leadership and communication are therefore pivotal.

2. Integration Complexity

Traditional roles and silos often resist the collaborative culture DevSecOps demands. Strong leadership and clear communication are key to overcoming this.

3. Security Concerns

Adding security to DevOps workflows requires careful planning, automation tools, and team training to avoid disruptions.

4. Limited Resources

Small enterprises often lack both the budget and expertise to establish DevSecOps. It is where the government helps by providing funds and appropriate training to narrow down this gap.

5. Regulatory Compliance

Industries that are governed heavily require compliance with regulations set by laws like GDPR, and HIPAA. Guidelines given by regulators need to be quite clear.

4. Results and Discussion

This helps explore DevSecOps and how it integrates into the DevOps lifecycle for the betterment of security and resilience in software development. By exploring these factors like cultural change, automation, resilience testing, real-world applications, and emerging trends, the following findings arise:

4.1. Cultural Change and Collaboration

A key finding is that organizations need to undergo a culture change. The development, security, and operations teams need to become one to attain shared security goals. To this end, barriers need to be removed between teams, open communication encouraged, and shared responsibility for security and resilience fostered. This needs to be anticipated by the organizations to check vulnerability before hitting into production. As automation tools may be used through which complex test scenarios would smash the

software as it might, in reality, break in the labs, in return testing stability along with security hence issues will also be found, thereby breaching and or downtime would never occur.

4.2. Resilience Testing and Automation

Alternatively, depending heavily on automation, security will be infused automatically into the DevOps pipeline in the process. This is through the automation of security testing, compliance checks, and incident responses that speed up the detection and resolution of vulnerabilities for security to remain consistent and efficient throughout the whole lifecycle.

4.3. Automating Security Processes

Real-world case studies show the practicability that an organization might adopt the use of DevSecOps practices. Organizations claim they have better system resiliency and improved operation efficiency while being secure in their workflows making this approach worthwhile.

4.4. Future Trends and Innovations

Emerging trends are the future of DevSecOps. Shift-left security, cloud-native security, zero-trust frameworks, and AI and machine learning-based practices have become the centerpieces for improving the security measures of organizations. The innovations enable these organizations to change their approach as threats change over time and increase their overall security posture.

4.5. Key Takeaway

Core integration of security with the DevOps lifecycle is indispensable for building security and resilience for software. All these- coordination, automation, real-world information, and adapting to future trends- allow modern challenges to organizations and make well-prepared systems for today's dynamic threat landscapes.

4.6. Discussion

DevSecOps provides many benefits but mainly in the enhancement of application security and how development processes could be streamlined. That is where integrated early security practices into the development cycle enable organizations to detect vulnerabilities early, which can reduce the chances of costly fixes at later stages in the development lifecycle. Some organizations have implemented the DevSecOps framework successfully and have leveraged automation tools, such as Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Software Composition Analysis (SCA). These tools help to identify and address vulnerabilities proactively so that problems are detected faster at lower costs.

One of the main findings of the literature review is that security checks must be automated at all stages of the software development lifecycle. Integrating security testing into the CI/CD pipeline is another important approach through which vulnerabilities are found earlier. This will help in overall system security, without slowing the development speed. In addition to this, following secure coding practices and training developers on the same can greatly reduce the possibility of introducing vulnerabilities. The final challenge is regulatory compliance, in particular, for sectors that are under rigid regulations like GDPR or HIPAA. DevSecOps frameworks need to conform to the security and compliance standards required by the regulations. There ought to be clear guidelines and standardized procedures in place for compliance purposes, yet with flexibility and agility through DevSecOps.

5. Conclusions

DevSecOps has proven to have remarkable potential in revolutionizing the approach of how security is integrated with the software development lifecycle. An organization can, therefore identify its vulnerabilities and reduce the cost incurred in the process of security testing at an early phase of development with the aid of incorporating security practices into every stage of development. Automation tools and the security testing platform ensure early identification and remediation of security problems so that they don't escalate further in development. However, the challenges related to the operation of change, integration complexity, resource constraints, and regulatory compliance need to be evolved for the successful implementation of DevSecOps. For that reason, the primary barriers may require solid leadership, effective communication, and proper training. In addition, the provision of necessary resources and training to small businesses will be considered critical in undertaking wide and broad adoption of DevSecOps practices across most industries. Emerging trends - machine learning and cloud-native security, zero trust models, or whatever will complement the evolution, will continue pushing the

boundaries, making DevSecOps more mature and powerful towards more robust frameworks. The immediate future of DevSecOps: it looks healthy, and prospects for these organizations look strong in terms of an efficient yet secure and strong approach to developing software. Existing challenges and possible ways of evolving DevSecOps for wider spread will need relentless research and innovative ideas. Therefore, DevSecOps is an innovative approach to the integration of security into the development of software. However, the implementation will require much deliberation over the organizational culture, tools, resources, and compliance with regulatory requirements. As this model continues to be adopted in organizations, it will progress further and make more sophisticated, streamlined ways of securing software applications.

References

1. Sharma, S.; Sharma, D. DevSecOps: Integrating Security with DevOps. Int. J. Comput. Appl. 2021, 179, 1–7. https://doi.org/10.5120/ijca2021917305
2. Li, X.; Yang, C. Adoption of DevSecOps: Challenges and Opportunities. In Proc. Int. Conf. Cloud Comput. and Security, 2020; pp. 235–242. https://doi.org/10.1007/978-3-030-41391-5_23
3. Soni, S. P.; Tiwari, P. DevSecOps: A Survey on Evolution, Tools, and Practices. In Proc. Int. Conf. Adv. in Comput. and Commun. Eng., 2021; pp. 218–226. Springer. https://doi.org/10.1007/978-3-030-44611-7_22
4. Fischer, M.; Grebe, J. Automating Security: Static and Dynamic Testing in DevSecOps Pipelines. IEEE Access 2020, 8, 214327–214341. https://doi.org/10.1109/ACCESS.2020.3033225
5. Debray, S.; Soni, P. SAST and DAST in DevSecOps: Improving Application Security. J. Comput. Security 2020, 28, 875–891. https://doi.org/10.3233/JCS-200679
6. Howard, M.; Lipner, S. The Security Development Lifecycle: A Process for Developing Demonstrably More Secure Software; Microsoft Press: 2020.
7. Sharma, M.; Gupta, A. Exploring the Role of Cloud-Native Security and Zero-Trust in DevSecOps. J. Cyber Security Technol. 2022, 6, 23–38. https://doi.org/10.1080/23742917.2022.2041234
8. Georgieva, I.; Petkov, K. Machine Learning for Secure DevOps: Current Trends and Future Directions. J. Softw. Eng. Appl. 2021, 14, 202–216. https://doi.org/10.4236/jsea.2021.145014
9. Rosenberg, J. D.; Andrews, C. T. DevOps and Security: A Symbiotic Relationship for Accelerated Software Delivery. Int. J. Softw. Eng. Appl. 2019, 10, 45–61. https://doi.org/10.5121/ijsea.2019.10304
10. Moeez, M.; et al. Comprehensive Analysis of DevOps: Integration, Automation, Collaboration, and Continuous Delivery. Bulletin of Business and Economics (BBE) 2024, 13, 653–661. doi: 10.61506/01.00253
11. Sandu, A. K. DevSecOps: Integrating Security into the DevOps Lifecycle for Enhanced Resilience. vol. 6, no. 1, 2021.
12. Zhou, X.; et al. Revisit Security in the Era of DevOps: An Evidence-Based Inquiry into DevSecOps Industry. IET Softw. 2023, 17, 435–454. doi: 10.1049/sfw2.12132
13. Iqbal, M.; et al. Enhancing Task Execution: A Dual-Layer Approach with Multi-Queue Adaptive Priority Scheduling. PeerJ Comput. Sci. 2024, 10, e2531. doi: 10.7717/peerj-cs.2531
14. Tezcan, Y. DevSecOps with Open Source Tools.
15. Olajide, A. O. DevSecOps: Integrating Security Testing into the Software Development Lifecycle. 2023, doi: 10.13140/RG.2.2.19207.20645
16. Akbar, M. A.; Smolander, K.; Mahmood, S.; Alsanad, A. Toward Successful DevSecOps in Software Development Organizations: A Decision-Making Framework. Inf. Softw. Technol. 2022, 147, 106894. doi: 10.1016/j.infsof.2022.106894
17. Gadani, N. N. Security Challenges in Cloud-Based Software Development: A DevSecOps Perspective. 2024.
18. Rajapakse, R. N.; Zahedi, M.; Babar, M. A.; Shen, H. Challenges and Solutions When Adopting DevSecOps: A Systematic Review. Inf. Softw. Technol. 2022, 141, 106700. doi: 10.1016/j.infsof.2021.106700
19. Prates, L.; Faustino, J.; Silva, M.; Pereira, R. DevSecOps Metrics. In Information Systems: Research, Development, Applications, Education, vol. 359, S. Wrycza, J. Maślankowski, Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 77–90. doi: 10.1007/978-3-030-29608-7_7
20. Akula, B. S. Vulnerability Management in DevSecOps. 2023, doi: 10.13140/RG.2.2.11943.97444
21. Tiedekunta, I. DevSecOps: Building Security into the Core of DevOps.
22. Bashiru, O.; Olufemi, O. G. An Enhanced CICD Pipeline: A DevSecOps Approach. Int. J. Comput. Appl. 2023, 184, 48, 8–13. doi: 10.5120/ijca2023922594
23. Schicchi, M. University of Turku, Department of Future Technologies.
24. Haider, S. W.; Ahmed, A.; Zubair, S.; Iqbal, M. W.; Arif, S.; Hamid, K. Fuzzy-Based Expert System for Test Case Generation on Web Graphical User Interface for Usability Test Improvement. vol. 42, 549–565, May 2023. doi: 10.17605/OSF.IO/H5ER9
25. Awais, M.; Tahir, M.; Farid, R.; Zubair, S.; Iqbal, M. W.; Hamid, K. Guide-A Tool to Detect the GUI Elements to Enhance the Efficiency of Testing. vol. 42, 55–69, Apr. 2023. doi: 10.17605/OSF.IO/2XBPZ
26. Rasool, N.; Khan, S.; Haseeb, U.; Zubair, S.; Iqbal, M. W.; Hamid, K. Scrum and the Agile Procedure's Impact on Software Project Management. Jilin Daxue Xuebao (Gongxueban)/J. Jilin Univ. (Eng. Technol. Ed.) 2023, 42, 380–392. doi: 10.17605/OSF.IO/MQW9P
27. Khan, S.; Ibraheem, I.; Ramay, S.; Khan, T.; Hassan, R. Identification of Skin Cancer Using Machine Learning. 2024

28. Salahat, M.; et al. Software Testing Issues Improvement in Quality Assurance. In 2023 International Conference on Business Analytics for Technology and Security (ICBATS), Mar. 2023; pp. 1–6. doi: 10.1109/ICBATS57792.2023.10111145
29. Akram, S.; Aqeel, M.; Iqbal, M. W.; Hamid, K.; Rafiq, S. Enhancing Software Quality through Usability Experience and HCI Design Principles. vol. 42, 46–75, Feb. 2023. doi: 10.17605/OSF.IO/MFE45
30. Hira, H.; Khan, M.; Afzal, S.; Zubair, S.; Atif, M.; Hamid, K. DevOps Methodology Impact on Software Projects to Lead Successes and Failures Through Kubernetes. Jilin Daxue Xuebao (Gongxueban)/J. Jilin Univ. (Eng. Technol. Ed.) 2022, 41, 11, Nov. 2022. doi: 10.17605/OSF.IO/D8YPH
31. Siddiquimohammad. Understanding DevSecOps the Easy Way. Medium, Feb. 11, 2025. Available online: https://medium.com/@siddiquimohammad0807/understanding-devsecops-the-easy-way-a24442c54c6e.
32. Kanstantsin, Z. Secure Change Management Process: On the Effectiveness of DevSecOps. Comput. Sci. Inf. Technol. 2022, 10, 37–51. doi: 10.13189/csit.2022.100401
33. Cloud. 10 Advantages of Using ArgoCD with Kubernetes. Successive Cloud, Feb. 11, 2025. Available online: https://successive.cloud/10-advantages-argocd-with-kubernetes/