# INTEGRATING SECURITY INTO THE SOFTWARE DEVELOPMENT LIFECYCLE: A DEVSECOPS PERSPECTIVE

**Mamura Abdumannopova**
Turin Polytechnic University in Tashkent
asretdinovamamura@gmail.com

**Abstract:** DevSecOps is an emerging paradigm that incorporates security practices into DevOps and agile software development methodologies. By automating security tasks and integrating tools at every stage of the continuous integration and continuous delivery (CI/CD) pipeline, DevSecOps helps organizations detect and remediate vulnerabilities early in the development lifecycle. This paper reviews key DevSecOps principles, including "shift-left" security and shared responsibility for security among development, operations, and security teams[1]. We discuss common tools and methods such as static and dynamic code analysis, dependency (software composition) scanning, and compliance-as-code that support a secure development pipeline[2][7]. The integration of security into automated CI/CD processes is examined, with examples of how vulnerability scanning and testing can be embedded at build and test stages[2][7]. Finally, we consider organizational and cultural factors, such as training and cross-team collaboration, that are critical to successful DevSecOps adoption[4][6].

## Introduction

DevSecOps (development, security, and operations) is an evolution of DevOps that integrates security practices into every phase of the software development lifecycle[1][4]. Traditionally, security testing was performed at the end of development as a separate phase. This "tacked-on" approach became a bottleneck as teams adopted agile and DevOps practices with weekly or daily release cycles[1][4]. In contrast, DevSecOps emphasizes automated security checks from the earliest stages of design and coding, ensuring that vulnerabilities are discovered and fixed promptly rather than after deployment[1][4]. For example, one guideline notes that embedding security measures in CI/CD pipelines "improves the capacity to discover and remediate vulnerabilities early," thereby reducing breach risk[4]. This shift-left approach makes security a shared responsibility among developers, security experts, and IT operations. As IBM describes it, DevSecOps "enables 'software, safer, sooner'" by automating secure delivery without slowing the development cycle[1].

## DevSecOps Principles and Tools

Key DevSecOps principles include integrating security early ("shift left"), automating security processes, and fostering shared responsibility for security[1][4]. In practice, this means that threat modeling and secure design are included from the planning phase, and that automated tests run continuously as code changes. For example, Atlassian recommends that teams "introduce security throughout the software development lifecycle," have all team members share responsibility for best practices, and "enable automated security checks at each stage of software delivery"[2]. In other words, security is treated as code: security controls and scans are defined in the same version controlled pipelines and enforced automatically.

A variety of tools support these principles. Static Application Security Testing (SAST) tools scan source code for patterns (e.g. injection flaws, XSS) during development or on each commit[2][5]. For instance, tools like SonarQube or SpotBugs can analyze code as part of a build. Dynamic Application Security Testing (DAST) tools test running applications (e.g. a deployed web app) to catch issues like SQL injection or broken authentication[5]. Dependency or Software Composition Analysis (SCA) tools (such as OWASP Dependency-Check or Snyk) scan third-party libraries and

container images for known vulnerabilities during the build phase[2]. Atlassian notes that important build-phase practices include "software component analysis, static application software testing (SAST) … [and] scanning [of] dependencies for any security vulnerabilities"[2]. In addition, modern DevSecOps pipelines often include secrets-scanning tools (to detect accidental credential leaks), container-image scanners (e.g. Trivy, Clair), and Infrastructure-as-Code scanners (for Terraform/CloudFormation) to enforce security in cloud setups. As the OWASP DevSecOps guideline describes, an ideal pipeline includes steps such as scanning code repositories for secrets, SAST, SCA, DAST, IAST (interactive testing), IaC scanning, and compliance checks[7]. By combining these tools, teams can cover different facets of security: static analysis catches coding issues early, while dynamic tests and runtime monitoring catch misconfigurations or logic flaws that only appear in running systems.

**Security Integration in CI/CD Pipelines**

DevSecOps integrates security directly into CI/CD pipelines. Each code commit or pull request can automatically trigger a suite of security checks. For example, a CI job might run a SAST scan on the new code and fail the build if critical issues are found. Similarly, a build-stage job can perform SCA to generate a Software Bill of Materials (SBOM) and flag known vulnerabilities in dependencies. After artifacts are built, a test-stage job can deploy to a staging environment and run DAST tools (like OWASP ZAP) against it. Atlassian recommends plugging security tools into the existing CI/CD pipeline so that "every commit and merge automatically triggers a security test or review"[2]. Research indicates that adding security measures into CI/CD "improves the capacity to discover and remediate vulnerabilities early"[4]. In line with this, NIST guidance emphasizes that DevSecOps should "automatically generat[e] security and compliance artifacts throughout the process, including software development, builds, packaging, distribution, and deployment"[3]. In practice, this means integrating scans and compliance gates in tools such as Jenkins, GitLab CI, or GitHub Actions so that no code progresses to production without passing the defined security policies. For example, container image scans (e.g. with Trivy or Clair) can be built into the deploy job, and infrastructure as-code scanners (e.g. Checkov) can be run before provisioning cloud resources. By baking in security at every stage of CI/CD, organizations enforce a consistent, automated security workflow.

**Organizational and Cultural Considerations**

DevSecOps is as much a cultural change as a technical one. It requires breaking down silos between development, operations, and security teams, and investing in training and communication. Organizations should foster a security-aware culture where "everyone involved with the delivery process is familiar with basic principles of application security"[1]. For example, IBM advises that all team members understand standards and practices such as the OWASP Top 10 and participate in threat modeling or secure code reviews. Likewise, DevSecOps often relies on roles like "security champions" within development teams to advocate for secure practices. As one expert put it, DevSecOps assumes "everyone is responsible for security" and empowers teams to make security decisions at scale[1]. RedHat similarly emphasizes that "DevSecOps is a culture," in which developers and security professionals collaborate closely and introduce security considerations earlier in the lifecycle[6].

Transitioning to DevSecOps can be challenging. Teams accustomed to separate security reviews may resist new processes and tools[6]. Common obstacles include the complexity of integrating tools into fast-release pipelines, legacy infrastructure, and organizational inertia. Studies have noted "cultural and organizational hurdles" as critical barriers to DevSecOps implementation[4]. To succeed, leadership must support the change through policy and incentives, and organizations often adopt a phased approach (e.g. starting with a few critical projects). Continuous education, executive

buy-in, and aligning security metrics with business goals help overcome resistance. Over time, as teams see the benefits (faster feedback on security and fewer late defects), DevSecOps practices become ingrained.

**Conclusion**

DevSecOps represents a mature approach to software security in the era of continuous delivery. By embedding automated security practices into the DevOps lifecycle, organizations can deliver features at speed without sacrificing security. This entails adopting principles such as shift-left testing and shared responsibility, and using a range of tools (SAST, DAST, SCA, and more) integrated into CI/CD pipelines. Equally important are the organizational changes: training developers in security, fostering collaboration, and evolving processes. When done correctly, DevSecOps enables the delivery of "software, safer, sooner"[1], ensuring that security is an inherent part of software delivery rather than an afterthought.

**References**

1. IBM,
"What is DevSecOps?"
IBM THINK, 2023. [Online]. Available:
https://www.ibm.com/think/topics/devsecops. [Accessed: Oct. 12, 2025].
2. K. Zettler, "The DevSecOps tools that secure DevOps workflows," Atlassian DevOps Blog, 2023. [Online]. Available: https://www.atlassian.com/devops/devops-tools/devsecops-tools. [Accessed: Oct. 12, 2025].
3. NIST National Cybersecurity Center of Excellence, "Secure Software Development, Security, and Operations (DevSecOps) Practices (NIST SP 1800-44A, Draft)", 2025. [Online]. Available: https://www.nccoe.nist.gov/projects/secure-software-devsecops. [Accessed: Oct. 12, 2025].
4. B. Leshchenko, B. Snisar, A. Stupak, and V. Osadchyi, "Integrating DevSecOps into the software development lifecycle: A comprehensive model for securing containerized and cloud native environments," in Proc. 2nd Workshop on Cybersecurity Providing in Information and Telecommunication Systems (CPITS-II), Kyiv, Ukraine, Oct. 2024, CEUR-WS, vol. 3826.
5. GitLab, "SAST vs DAST," GitLab DevSecOps Guide, 2024. [Online]. Available:
https://about.gitlab.com/topics/devsecops/sast-vs-dast. [Accessed: Oct. 12, 2025].
6. A. Kohgadai, "Building a DevSecOps culture and shifting security left," Red Hat Developer Blog, Aug. 24, 2021. [Online]. Available: https://www.redhat.com/en/blog/building-devsecops culture. [Accessed: Oct. 12, 2025].
7. OWASP Foundation, "OWASP DevSecOps Guideline," 2023. [Online]. Available:
https://owasp.org/www-project-devsecops-guideline. [Accessed: Oct. 12, 2025].