**DevOps and DevSecOps**

[DevOps](https://www.geeksforgeeks.org/how-devops-works/) combines “development” and “operations” to describe a collaborative or shared-responsibility model for building applications. It views the work performed by the software development and IT operations teams as a single process.

The DevOps philosophy promotes closer collaboration and constant communication between the different teams within an organization. In its narrowest sense, DevOps refers to iterative processes involved in application development, automation, and the deployment and maintenance of programmable infrastructure.

DevOps also refers to a cultural change, such as building trust between system admins and developers and aligning technology projects with business objectives. DevOps can transform an organization’s software delivery pipeline, job functions, tools, and practices.

[DevSecOps](https://www.geeksforgeeks.org/development-security-and-operations-devsecops-engineer/) is a software development management approach introducing security to the DevOps equation. It creates an automated Continuous Delivery (CD) pipeline by combining development, operations, security, and infrastructure as code (IaaS).

The primary purpose of DevSecOps is to automate, manage, and enforce security throughout the [software development lifecycle (SDLC)](https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/). It requires monitoring and applying security at each pipeline stage, including planning, building, testing, delivery, deployment, operations, and monitoring. Implementing security practices at all stages of the development process allows organizations to achieve Continuous Integration (CI), lower compliance-related costs, and accelerate software delivery.

DevSecOps requires all teams and employees to take responsibility for security from the start and perform their tasks without compromising security.

**Importance of DevSecOps:**

Organizations today rely on complex on-premises, cloud-based, and hybrid environments to support IT operations. Adding to this complexity is the constant creation of new applications and updates. Many organizations use cloud containers and microservices to develop applications in-house.

Whenever an employee creates or modifies a component or asset connected to the Internet, a misconfiguration or new vulnerability could expose the application to attack. Additional complexities that increase the risk include accelerating development, automating parts of the application delivery process, and splitting applications into microservices. Developers often make small mistakes, leaving assets vulnerable to cyberattacks.

Software engineering teams also use tools to automate tasks like configuring and maintaining containers, servers, image registries, and code repositories. All these components can introduce security vulnerabilities.

While standard DevOps workflows deliver tangible business value, they are also a significant source of risk. Therefore, the security focus of DevSecOps is crucial. Securing the software development environment should be an integral part of the development process, not an afterthought.

**DevOps and  DevSecOps:**

DevOps is a broad concept encompassing various organizational strategies. The core of DevOps is the shared responsibility between traditionally separate teams. It originated as a general paradigm with common practices but has now become a well-defined workplace culture and development process. Organizations that adopt a shared responsibility approach to development and operations can produce faster iterations and release more successful applications. DevSecOps extends this philosophy by incorporating security goals and practices into the overall business objectives.

DevSecOps is a natural evolution of DevOps, not a separate concept. Successful DevOps teams, already used to DevOps practices, can approach DevSecOps as the logical next step in the DevOps adoption process.

While the initial objective of DevOps was to create business value in a seamless development workflow (from build to production environments), many standard DevOps tools and practices neglect security, focussing on fast releases. The result was often security bottlenecks, where traditional security pipelines were too slow for the faster DevOps process.

Many organizations left security to post-production or even an external team, resulting in slow security feedback loops.

**Difference Between DevSecOps and DevOps:**

| **Parameters** | **DevOps** | **DevSecOps** |
| --- | --- | --- |
| **Culture** | Promotes a work culture or shared responsibility for development and operations. | Emphasizes security, extending the culture of shared responsibility. |
| **Treatment of security** | Usually implements security at the end of the SDLC. | Introduces security into the continuous integration and continuous development (CI/CD) pipeline. |
| **Security tools** | Combines modern DevOps pipelines with traditional security methods. | Requires teams to embrace new security tools and techniques. The DevOps process must include security tools and controls from the beginning, adapting security to the CI/CD workflow. |
| **Efficiency** | Often results in security bottlenecks and technical debt due to slow feedback loops. | Reduces vulnerabilities in production, minimizing the costs of addressing security issues and bugs. It enables scalability without compromising security, placing secure code as a core DevOps objective. |
| **Automation** | Automates development processes but relies on a human team to handle security | Brings security to all stages of the development and delivery process, embracing automation to accelerate security tasks. |

Thus, the main difference between DevSecOps and DevOps is the emphasis on security—it extends the DevOps work culture to promote shared responsibility for security practices. Instead of implementing security at the end of the SDLC, DevSecOps introduces it into the continuous integration and continuous development (CI/CD) pipeline.

A successful DevSecOps strategy requires teams to embrace new security tools and techniques rather than trying to combine traditional security methods with modern DevOps pipelines. The DevOps process must include security tools and controls from the beginning, adapting security to the CI/CD workflow.

This approach reduces vulnerabilities in production, minimizing the costs of addressing security issues and bugs. It enables scalability without compromising security, placing secure code as a core DevOps objective. DevSecOps brings security to all stages of the development and delivery process, embracing automation to accelerate security tasks.

**Transitioning from DevOps to DevSecOps:**

The first step towards DevSecOps is to familiarize team members with the ideas behind security. Once everyone is on board with the adoption process, the organization can start making changes to the development process. All employees must recognize the benefits of implementing application security from the start of the SDLC.

Although there are many security testing methods, it can be difficult to determine which method is best for a specific organization or project. Here is an overview of the basic testing techniques:

* **Static Application Security Testing (SAST**)—examines code to identify weaknesses.
* **Dynamic Application Security Testing (DAST)**—involves administrators identifying vulnerabilities and security gaps.
* **Interactive Application Security Testing (IAST)**—combines SAST and DAST to enable software instrumentation to screen applications.
* **Runtime Application Self-Protection (RASP)**—leverages real-time app data to enable the automatic identification and remediation of attacks without an administrator’s involvement.
* **Software Composition Analysis (SCA)**—automatically identifies third-party and open libraries in an application, identifies known vulnerabilities, and notifies users of available updates and patches.

Code quality assessment is an important part of DevSecOps. It ensures that code is normalized and stable, making it easier for teams to keep it secure in the future. Organizations should regularly educate their developers to promote secure coding practices and ensure they implement all code changes consistently.

Another aspect of transitioning to DevSecOps is to set up protections for applications running across distributed infrastructure rather than relying on a security perimeter. This implicit approach to security is easier to maintain in fast-growing and changing environments.