GitHub

The enterprise architect's guide to DevSecOps

Introduction

Enterprise DevOps is here to stay. But enterprise architects have long understood what DevOps leaders are just discovering—security, not only shipping, is a shared responsibility.

Today, operations teams use collaboration, automation, and containers to speed up software delivery. While these DevOps best practices have helped them find new ways to build faster, old security practices still slow many organizations down.

Enter DevSecOps. DevSecOps brings IT security into development and operations teams to ensure that security is a priority at every step of the software development lifecycle. With a few changes, your organization can ship better, more secure software—without delays or increased costs.

Why DevSecOps?

Lower security costs

DevSecOps includes all the DevOps best practices high-performing teams live by, with the security large organizations require. By building security into your DevOps pipeline, it's possible to find vulnerabilities before they're ever released—and easier and less expensive to remediate them.

More effective teamwork

Just as developers and operations are both responsible for reliability and quality in DevOps, DevSecOps makes security a team effort, not a final step. Developers, operations, and security teams work together to keep applications secure from the first line of code to final production.

Policy-driven automation

A good DevSecOps program also increases confidence in your organization's entire software delivery process. Automated checks implement security in a policy-driven way, rather than as a set of confusing manual tools that slow development down for everyone.

THEN:

Siloed security

Testing just before deployment

Static testing and dynamic testing happened at the end of the delivery cycle, right before release.

Separate security expertise

Development, IT operations, and security teams worked independently.

Manual security testing

Organizations deployed less often and ran security checks individually, as needed.

NOW:

DevSecOps

Testing from idea to production

Static and dynamic testing happens alongside secure coding practices, quality gate checks, and security vulnerability fixes.

Shared security expertise

Developers, IT operations, and security teams all follow shared security guidelines in their work.

Automated security testing

Organizations deploy more frequently and add automated security checks to their CI/CD pipeline.

Three DevSecOps tips to get started

1 Use a shared, safe platform for collaboration

Like DevOps, DevSecOps depends on and ends with collaboration. A shared platform helps development, IT operations, and security teams build together and standardize how they work. Prioritize platforms with built-in security so your entire organization can share best practices, find and reuse code, and collaborate from the start.

GITHUB TIP

Good security begins at sign in. When you find the right collaboration platform, it should also support identity management features like two-factor authentication, single sign-on, automatic organization syncs, and more.

2 Secure your SDLC from end to end

Up to 99 percent of recently released applications contain open source code—meaning open source dependencies are already part of your codebase.* Integrate code security tools into your CI/CD pipeline that can proactively identify security vulnerabilities in both open and internal source code.

*2019 Open Source Security and Risk Analysis Report

GITHUB TIP

Open source software is everywhere. Automated security tools like LGTM variant analysis, WhiteSource, and Snyk can make it easy to find and eliminate bugs and vulnerabilities your team can't track by hand.

3 Track security after production

Security doesn't end once code is committed—and neither does your DevSecOps pipeline. After deployment, keep code and customers safe by continuously monitoring for vulnerabilities. Look for tools that can track and update vulnerable dependencies post-launch, before would-be hackers can take advantage.

GITHUB TIP

While security vulnerability alerts make projects safer, industry data shows that more than 70 percent of vulnerabilities remain unpatched after 30 days—and many up to a year. Use integrations that don't just identify vulnerable dependencies, but fix them automatically.

Questions about secure software development?

We can help.

Learn more at
github.com/enterprise