BLACK DUCK®

HOW TO NAVIGATE THE INTERSECTION OF DEVOPS AND SECURITY

A BRIEF HISTORY OF DEV VS. OPS

Traditionally, the software development process was split into two distinct phases and among three separate teams, and it could take months, if not years, to deploy an application. Each of the three teams has different responsibilities, often with little communication between them. In the first phase, the development team writes and compiles code, then releases it to the IT operations team. In the second phase, the operations team makes sure that the code runs and remains stable in production environments. Then, the third team, IT security, which typically works closely with operations, ensures the software is secure before it is deployed.

The functional responsibilities of these teams often conflict with each other. Further, this inefficient division of labor and focus often leads to poor-quality software, slow deployments, long lead times to patch and fix vulnerabilities, and frustrated customers. In today's world of rapid software releases, this kind of development cycle just doesn't cut it. Enter DevOps.

"DevOps adoption means that you are willing to change fast, develop fast, test fast, fail fast, recover fast, learn fast, and push the product/features fast to the market."

—Pavan Belagatti

What is DevOps?

DevOps bridges the two worlds of development and operations. The goal is to increase the efficiency, speed, and security of software development and delivery, thus enabling organizations to serve their customers effectively and increase their competitive edge. DevOps is not a one-and-done practice; it's a way of working that has the primary aim of maximizing the efficiency of the software development process by breaking down silos. It combines philosophies, technical practices, and tools to optimize the software development life cycle (SDLC) and create cross-disciplinary teams that work together toward a single goal: to build, test, and release software faster and more reliably.

Some organizations merge development and operations into one DevOps team of engineers that works across the application life cycle, building out development, test, deployment, and operations skills. Many organizations also integrate quality assurance and security teams into the process throughout the application life cycle.

DevOps advocates automating processes and using tools to increase transparency and communications across teams, and improve applications faster and more reliably. By making a secure final product everyone's job, DevOps builds trust between groups, leading to faster software releases and the ability to solve critical issues quickly and manage unplanned work effectively—creating a culture of continuous improvement and innovation that increases competitive advantage.

Benefits of DevOps

- **Speed.** Faster innovation and ability to adapt to evolving market conditions.
- **Rapid delivery.** Accelerated product improvements and faster time to market.
- **Reliability.** Quality application updates and infrastructure changes, while maintaining higher product quality and delivering an optimal user experience.
- **Improved collaboration.** Shared responsibilities and combined workflows reduce inefficiencies and improve productivity.
- **Security.** Automated, integrated security testing tools enable faster identification and mitigation of risks.

"In the DevOps ideal, developers receive fast, constant feedback on their work, which enables them to quickly and independently implement, integrate, and validate their code, and have the code deployed into the production environment."

—Gene Kim, "The DevOps Handbook"

DEVOPS PRINCIPLES: THE THREE WAYS

The blog post "The Three Ways: The Principles Underpinning DevOps" describes the values and philosophies that frame the processes, procedures, and practices of DevOps.

The first way: Flow/systems thinking

The first way focuses on the performance of the entire system, rather than looking at a specific department or silo. It is centered on how IT can deliver business value through a rapid flow of work from development to operations, and on to delivery to the customer. It concentrates on removing constraints, never passing down a known defect, and looking for ways to speed up the flow. This involves four tasks.

- Make work visible; that is, find a way to visualize the work so you can see the problems and resolve them
- Reduce both the batch sizes of the work (the requirements, code, and tests in each work item) and work intervals
- Build quality into the process by preventing defects from being passed downstream
- Keep optimizing for global goals

The second way: Amplify feedback loops

The second way is designed to shorten and amplify the feedback loop at all stages, from right (operations) to left (development). This embeds knowledge where it is needed and empowers teams to boost feedback, avoid duplicating problems, and enable faster detection and recovery. The second way is about

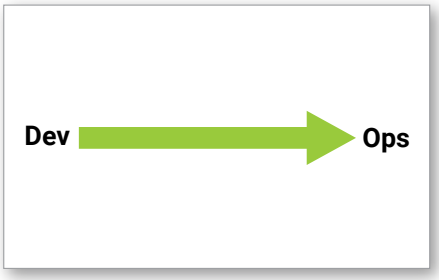
- Seeing problems as they occur
- Swarming to solve problems and build knowledge
- Pushing responsibility for quality closer to the source (i.e., further left in development)

The third way: Culture of continual experimentation and learning

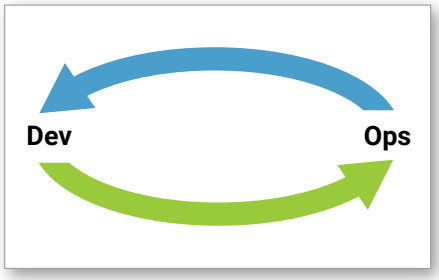
The third way creates a collaborative, high-trust work culture that fosters controlled experimentation and risk-taking, which, thanks to fast feedback loops, facilitates organizational learning from both achievements and failures. Outcomes include

- Allocating time for improving daily work
- Transforming local discoveries into global improvement
- Introducing faults into the system to increase resilience

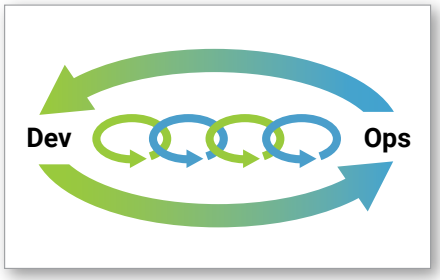
1. Flow



2. Feedback loops



3. Experimentation and learning



DEVOPS PRINCIPLES: SECURITY

The final step to achieving DevOps is integrating security, change management, and compliance controls into the day-to-day jobs of development and operations teams. Doing so makes security a part of every role in the SDLC, instead of limiting that responsibility to the security team. Building automatic security controls into the processes and tools that development and operations teams already use can go a long way toward a successful DevOps implementation. Incorporating security earlier in the process shortens feedback loops and decreases complexity, allowing engineers to fix security or compliance issues faster and more easily.

It costs six times more to fix a bug found during implementation than one identified during design.

—IBM Systems Science Institute

WHAT IS DEVSECOPS?

Dr. Tapabrata Pal, vice president of architecture at Fidelity Investments, described the process of integrating security into all stages of the SDLC as "DevOpsSec." Today, we call it DevSecOps or secure DevOps. The practice requires cultural and practical changes including integrating security into defect-tracking and postmortems, shared source code repositories and services, and the deployment pipeline. It also means ensuring the security of the application and the entire software supply chain.

Integrating security into defect-tracking and postmortems

Ensure security visibility and effective risk prioritization by tracking security issues in the same work-tracking systems that your development and operations teams already use. Then, take steps to prevent your team from repeating the same problems by performing regular reviews or providing security education and remediation guidance to developers.

DevSecOps tools

Most common work-tracking system

- Jira issue- and project-tracking software

Source code management tools

- Git (e.g., GitHub, GitLab)
- Bitbucket
- Mercurial

Binary repositories

- Artifactory
- Nexus

Continuous integration tools

- Jenkins
- Travis
- TeamCity

Continuous delivery/ deployment tools

- Jenkins
- XebiaLabs
- GoCD Nexus

Integrating security into shared source code repositories and services

Use source code management tools to create a repository of security-approved libraries that satisfy specific security objectives. This repository can also contain packages and builds approved for use in development (such as secure versions of OpenSSL with correct configurations), toolchains, the deployment pipeline, and standards.

A best practice when establishing a security program is performing an automated analysis of artifacts as they are checked into repositories, and a periodic analysis of the contents of the repository itself. This helps ensure consistent security risk awareness as software gets pushed downstream and as new vulnerabilities are disclosed.

Integrating security into the deployment pipeline

Automate as many security tests as possible to run alongside other automated tests in your deployment pipeline—at every major code commit, and even at very early stages. The goal is to provide short feedback loops that notify development teams of any potential security issues in code commits, and alert operations teams to any new risks to previously deployed artifacts. This allows teams to detect and correct security problems quickly as a part of daily work instead of waiting until the end of the SDLC when fixes are often complex, time-consuming, and expensive. The easiest way to achieve this is by using continuous integration tools and continuous delivery/deployment tools.

Ensuring the security of the application

Automate tests to run continuously in your deployment pipeline, instead of performing unit or functional tests manually. This is critical for the QA team, which will want to include static and dynamic analysis, software composition analysis (SCA), interactive application security testing (IAST), container scanning, and more. Many of these testing processes can be part of a continuous integration (CI) or continuous delivery/deployment (CD) pipeline, effectively creating a security pipeline that runs parallel to CI/CD pipelines to avoid unnecessary impediments to workflows. There should also be defined policies for security processes and risk tolerance to ensure security gates are automatically enforced only where appropriate.

"Seventy-nine percent of organizations admit to pushing application changes with known organic vulnerabilities."

—ESG

Ensuring the security of the software supply chain

The software supply chain reflects the combination of proprietary code, open source components, and third-party artifacts that compose modern software. DevOps teams must consider that applications inherit both the functionality and security vulnerabilities of open source code. Being aware of known vulnerabilities in open source helps developers choose which components and versions to use before they structure entire projects around vulnerable artifacts. Components should be analyzed during development within the IDE, and after the build when other dependencies are resolved into the compiled application.

Additionally, it's important to analyze third-party software and compiled binaries for vulnerabilities. This may include SCA of the binary itself and analysis of the application in a running state in a secure test environment. Some IAST technologies can run in parallel to standard functional tests, providing security risk insight without adding test cycles outside those that are already part of established workflows.

Jonathan Knudsen, former senior security strategist at Synopsys (now Black Duck), points out,"If you think about how software is made and deployed and maintained, it's a whole supply chain. And it starts when you're designing software or you're thinking about new features. In the design phase, you have to be thinking about security. You have to do threat modeling or architectural risk assessments, so before you write any code, you're just thinking about how it's going to work, and what it's going to do—and how it could be attacked."

WHAT TOOLS ARE NEEDED TO ACHIEVE DEVSECOPS?

Ideally, application security (AppSec) streamlines testing and remediation throughout the entire project workflow, from development to deployment, and distributes the responsibility for security across the development, operations, and security teams. Automation is necessary to achieve this—seamlessly integrating tools removes complexity from the SDLC, making the process faster and more efficient.

58% of organizations say application security is their top security investment priority.

70% of organizations are using 11 or more automated application security testing tools.

Secure code

Addressing security in the code without changing the workflow and before pushing vulnerabilities and risks downstream is a critical step to achieving DevSecOps. Developers need tools that provide actionable and accurate remediation guidance to address security problems early in the SDLC, when they are easier to fix.

IDE plugins

IDE plugins address security defects in real time as developers code the software. They offer visibility into open source dependencies and licenses, API calls, and infrastructure-as-code files. By receiving real-time feedback while writing code, developers can address security issues early and reduce the risk that a vulnerability will make it into production.

SAST

Static application security testing (SAST) analyzes application source code, including byte code and binaries, for coding and design conditions that may indicate security vulnerabilities. SAST solutions analyze applications at the source code level—without executing code—and find critical defects and security weaknesses so they can be fixed before release. It's essential to integrate an automated SAST solution into the SDLC to continuously identify quality defects and potential security vulnerabilities as code is written.

SAST TOOLS

Look for a SAST tool that

- Provides deep, full path coverage
- Supports thousands of developers
- Analyzes projects exceeding 100 million lines of code
- Integrates with your key development tools and CI/CD system

Continuous testing and verification

Always be testing: Continuous testing and verification is another critical element of DevSecOps. By handling large volumes of applications and releases, incremental scanning reduces costs, improves scalability, and addresses the needs of various applications, products, and stages in the SDLC.

SCA

Software composition analysis (SCA) statically analyzes source code or binaries. SCA tools identify open source components and any known associated security vulnerabilities. Often, SAST tools alone cannot identify these open source vulnerabilities, so it is important to incorporate SCA for a secure DevOps pipeline.

Since open source can be pulled into code in many ways, powerful SCA solutions use multiple scanning methodologies to identify all the components (full or partial) and their dependencies. Some tools provide remediation guidance for developers as well. Like SAST, SCA tools should integrate easily into your development toolkit, most commonly in a CI/CD system.

Automation

Modern software development is a complex computing model that relies on breaking down applications into smaller bits, such as serverless functions, microservices, and APIs, to enable faster and more resilient design.

IAST

Interactive application security testing (IAST) analyzes running application behavior during the testing phase, working in the background during manual or automated functional or security tests. Unlike DAST, IAST solutions use code instrumentation to observe application behavior and data flow, identifying vulnerabilities and providing developers with the information needed to pinpoint, prioritize, and remediate them. These capabilities make IAST a good solution for CI/CD environments, where speed and automation are a priority.

The culture of DevSecOps

Adopting a DevOps culture can't happen overnight. It is a process of creating alignment between your development, operations, and security teams, and training is a key part of it.

High-profile breaches occur on a weekly (sometimes daily) basis, and they serve as a reminder of how important software security is, but your security and QA teams can't resolve all security issues alone. Every employee in your development organization must take responsibility for security.

Software security is a complicated topic. To be effective and engaging for your employees, security training must deliver information relevant to their role or project. You don't want the same type of security training for your developers as for QA engineers, architects, and so on. Measuring the success of the training is also essential to building your team's security competency and helping your organization achieve security compliance.

BRINGING IT ALL TOGETHER FOR A SUCCESSFUL DEVSECOPS CULTURE

Balancing security and compliance with short delivery timelines isn't easy. The time pressure can lead to pushing insecure code into production. As application security shifts left into development, teams need guidance and structure to balance the competing goals of speed and security. That guidance comes from a DevSecOps culture that provides the tools, workflows, and technologies to build a modern application software development process.

The DevSecOps approach enables teams to build secure code from the ground up. It provides a process that incorporates continuous testing and verification, automates as much as possible with industry-leading AppSec testing tools, and orchestrates the right tests at the right time throughout the SDLC, without getting in the way of the development pipeline.

BENEFITS OF IAST

- **Automation:** Runs without human intervention
- **Accuracy:** Provides high-quality findings
- **Actionable results:** Pinpoints the exact line of code where a vulnerability was found
- **Integration:** Fits easily into the CI/CD process
- **Inclusive:** Tests web applications, web APIs, and microservices

DevSecOps is a big undertaking, so the best advice is to start small. Simple changes in the process can deliver quick wins—take advantage of the continuous integration, continuous delivery, and continuous deployment tools that your development and operations teams are already using. Then integrate SAST and SCA to gain visibility into and control over your open source code. Finally, add intelligent orchestration tools that enable full integration across the SDLC.

Learn more about building security into DevOps

1. Gene Kim, The Three Ways: The Principles Underpinning DevOps, ITrevolution.com, August 22, 2012.
2. Maurice Dawson, Darrell Norman Burrell, Emad Rahim, Stephen Brewster; Integrating Software Assurance into the Software Development Life Cycle (SDLC), Researchgate.net, January 2010.
3. Dave Gruber, Cracking the Code of DevSecOps, ESG, June 2021.
4. ibid.
5. 2022 Edgescan Vulnerability Statistics Report, Organizations Take an Average of 60 Days to Patch Critical Risk Vulnerabilities, PRNewswire.com, March 7, 2022.
6. Verizon, 2022 Data Breach Investigations Report, 2022.

ABOUT BLACK DUCK

Black Duck® meets the board-level risks of modern software with True Scale Application Security, ensuring uncompromised trust in software for the regulated, AI-powered world. Only Black Duck solutions free organizations from tradeoffs between speed, accuracy, and compliance at scale while eliminating security, regulatory, and licensing risks. Whether in the cloud or on premises, Black Duck is the only choice for securing mission-critical software everywhere code happens. With Black Duck, security leaders can make smarter decisions and unleash business innovation with confidence. Learn more at www.blackduck.com.