Shift Left DevOps

What Is Shift Left DevOps?

The term “shift left” refers to the efforts of a DevOps team to guarantee application security at the earliest stages in the development lifecycle, as part of an organizational pattern known as DevSecOps (collaboration between development, security, and operations).

To shift left means to move a process to the left on the traditional linear depiction of the software development lifecycle (SDLC). There are two common subjects of shift left initiatives in DevOps: security and testing.

In this article, you will learn:

What Does Shift Left Mean for Testing and Security?

Shift left testing

Traditionally, application testing was implemented during the last phases of development, before being sent to security teams. If an application did not meet quality standards, did not function properly, or otherwise failed to meet requirements, it would be sent back into development for additional changes. This caused significant bottlenecks in the SDLC and was not conducive to DevOps methodologies, which emphasize development velocity.

Shift left testing makes it possible to identify and fix defects much earlier in the software development lifecycle. This streamlines the development cycle, dramatically improves quality, and enables faster progression to later stages for security analysis and deployment.

Shift left security

Until recent years, security testing was implemented at the end of the development cycle, following application testing. At this stage, security teams would perform various types of analysis and security testing, such as static analysis (SAST) and dynamic analysis (DAST).

The results of security testing would either permit the application to proceed for deployment into production, or reject the application and pass it back to developers for remediation. This resulted in long delays in development or increased risk of releasing software without necessary security measures.

To shift security left means to implement security measures during the entire development lifecycle, rather than at the end of the cycle. The goal of shifting security left is to design software with security best practices built in, and to detect and fix potential security issues and vulnerabilities as early in the development process as possible, making it easier, faster, and more affordable to address security issues.

Why Shift Left Testing?

By performing testing earlier in the development cycle, developers can catch problems early and fix them before they reach the production environment. Because issues are discovered earlier, developers do not waste time applying workarounds to flawed implementations, and operations teams are not tasked with maintaining a faulty application in production. Developers can identify the root cause of issues and change application architecture or modify underlying components to improve application quality.

Another major advantage of shifting testing left is that testers are involved in the whole cycle, including the planning phase. Developers take on a secondary role as testers, becoming proficient in automated testing technologies and running tests as part of their day-to-day work. Testing becomes part of the “DNA” of the development organization, ensuring software is designed from the ground up with quality in mind.

Why Shift Left Security?

Before the advent of agile development practices and cloud computing, developers would request infrastructure from IT and receive a server weeks or months later. Over the past two decades, IT has shifted left. Today development infrastructure is fully automated and operates on a self service basis:

- Developers can provision resources to public clouds such as AWS, GCP, or Azure without involving operations or IT staff
- Continuous integration and continuous deployment (CI/CD) processes automatically set up testing, staging, and production environments in the cloud or on-premises and tear them down when they are no longer needed
- Infrastructure-as-Code (IaC) is widely used to deploy environments declaratively, using tools like Amazon CloudFormation and Terraform
- Kubernetes is everywhere, enabling organizations to provision containerized workloads dynamically using automated, adaptive processes

This shift has tremendously improved development productivity and velocity, but also raises serious security concerns. In this fast paced environment, there is little time for post-development security reviews of new software versions or analysis of cloud infrastructure configurations. Even when problems are discovered, there is little time for remediation before the next development sprint begins.

The Technology Driving Shift Left Security

DevOps organizations realized that they must also shift security left to avoid introducing more security risks than security and operations teams can manage. This movement is known as DevSecOps, and uses a variety of tools and technologies to close the gap and enable rapid, automated security assessment as part of the CI/CD pipeline:

- **Static Application Security Testing (SAST)** is used to scan source code for known weaknesses and insecure coding practices. In DevSecOps, this testing is typically integrated into developers’ development environments for immediate security risk feedback.
- **Software Composition Analysis (SCA)** analyzes software to detect known software components, such as open source and third-party libraries, and identify any associated vulnerabilities. SCA complements SAST by finding vulnerabilities not detectable by scanning source code.
- **Dynamic Application Security Testing (DAST)** scans applications in runtime, prior to deployment into production environments. This enables an outside-in approach to testing applications for exploitable conditions that were not detectable in a static state.

- **Web Application Firewalls (WAF)** monitor traffic at the application level and detect potential attacks and attempts to exploit vulnerabilities. WAFs can be configured to block certain potential attack vectors even without remediating the underlying software vulnerabilities.
- **Container image scanning tools** can continuously and automatically scan container images within the CI/CD pipeline and in container registries, prior to deployment into production environments. This enables identification of vulnerabilities or unsafe components, and provides remediation or mitigation guidance directly to developers and DevOps teams.
- **Cloud Security Posture Management (CSPM)** solutions identify misconfigurations in cloud infrastructure that could leave potential risks and attack vectors unchecked. CSPM solutions can recommend or automatically apply security best practices based on an organization's internal policies or third-party security standards.

Learn more in our guide to DevSecOps tools ›

Cultural Elements of Shift Left Security

While these solutions each provide valuable capabilities at each stage in the SDLC and CI/CD pipelines, these technologies alone are not enough to fully secure cloud native applications. The essential component of shift left security is people.

Just like DevOps was the methodological response to operational inefficiencies between development and operations teams, DevSecOps requires representation of security within this process. It is giving rise to a new generation of developers with additional security responsibilities and an increased proficiency in addressing risks.

Developers must be aware of security at all stages of the development lifecycle—whether they are selecting a package during project planning, writing a function, running a test, or deploying an environment. They must consistently account for security risks and take appropriate, informed action to preclude them before they advance into production.

All contributors must leverage automated tooling and deep integration of security resources to make security for cloud native applications practical and feasible at high development velocities—and developers must be willing and able to use it.

A Process for Implementing Shift Left Security

Define Common Goals

Shift left requires a cultural and organizational change. To facilitate this change, all team leaders should make decisions together. This helps ensure that any process or tool introduced into the development cycle has broader implications on all stakeholders. Organizations can promote this collaboration by encouraging leaders and teams to discover commonalities and align success criteria.

In addition, teams should discuss risks and real-world impacts of existing development and testing methodologies and establish a clear scope and definition of the shift to a new unified DevSecOps methodology. Teams can whiteboard current security challenges and then outline how shifting left can help. Once this information is gathered, teams can assess the pros and cons of the new methodology.

When shifting left, organizations need to reorganize processes. Ideally, the new processes are designed to benefit the organization as a whole. By including all stakeholders in these discussions, the organization can facilitate an effective and efficient transition.

Understand the Software Supply Chain

In order to shift security to the left, you need to know where and how your organization's software is created. For many organizations, this includes a mix of in-house and third-party developers and software vendors. This software supply chain adds complexity when architecting a comprehensive DevSecOps and shift left security program.

It is important to fully assess your software supply chain, and understand that your security risk posture is, to a large degree, subject to the security proficiency of others. Take this into account and establish the tools, standards, and practices necessary to safeguard your organization from such risk exposure.

Automate Security Processes

A shift left process typically involves the introduction of new technologies into the pipeline and retiring technologies that are no longer needed. Tools play a major role in DevOps and DevSecOps pipelines, facilitating collaboration, automation, and generally supporting the efforts of various teams. Choosing the right tools can help teams establish security practices during all stages of the lifecycle in a quick and timely manner.

Here are several important tools that can help automate security and integrate it into the development pipeline:

- **Continuous Integration (CI) tools**—help teams push code to production without delay, based on pre-defined triggers, and can initiate automated tests through integration, which can include security tests.
- **Test automation tools**—help automate application functionality testing, which can include tests for application vulnerabilities.
- **Issue tracking tools**—once security risks are detected, teams can be automatically notified of the issues, which can be triaged accordingly and relevant information can be provided to accelerate remediation.
- **Container image and serverless function scanning**—containers and functions are commonly used in cloud native applications and DevOps workflows. Scanning technologies can be configured to automatically analyze these artifacts as they are checked into container registries and function stores. This ensures analysis before they are deployed into production or used in other development activities.

Train Development Teams in Secure Coding

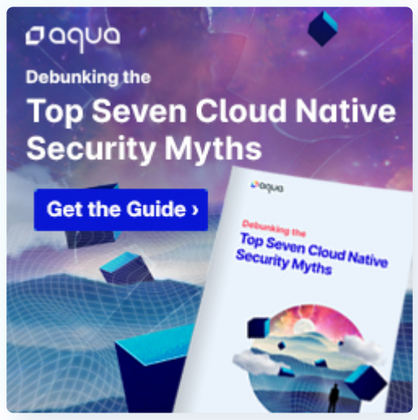
As the population of career developers and software engineers grows, the disparity of security risk awareness and secure coding proficiency is becoming more apparent. Many developers were not trained in secure coding practices or lack adequate resources to identify security shortcomings, and research shows that a large proportion of developers do not code securely and are not confident in their organization's security practices.

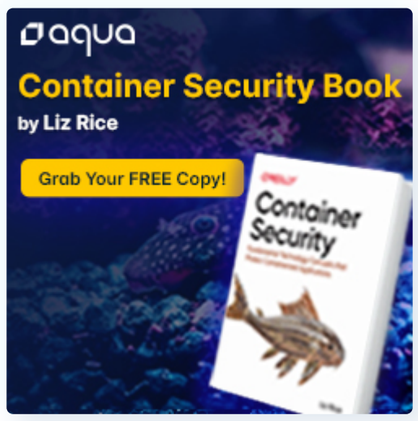
This human aspect is often neglected, as many organizations focus more on tools and organizational models. However, developer education is a critical part of the journey to shift left security. To be successful, an organization must create awareness and impart the relevant skills to the people who actually do the coding.

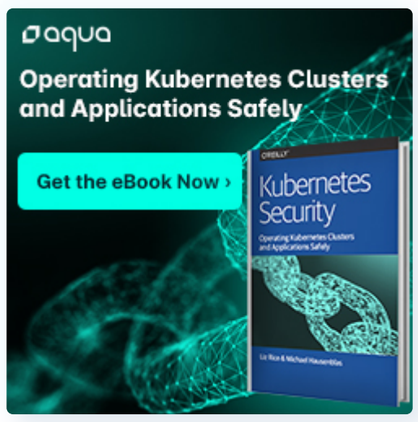
DevSecOps for Cloud Native with Aqua Security

Aqua Security provides solutions to detect, identify, and prevent security risks in cloud native applications from development and into production. Aqua integrates directly, and via API, with resources used for end-to-end DevSecOps, enabling security risk analysis at build, in container registries, in function stores, and at runtime.

Aqua takes a three-pronged approach to DevSecOps by empowering organizations to secure applications and containers as they are built, to ensure secure configuration of the cloud environments in which those artifacts run, and to ensure security at runtime.







Aqua Blog
All about cloud native and security ›