aqua

A DevSecOps Guide

In this guide, you will learn about:

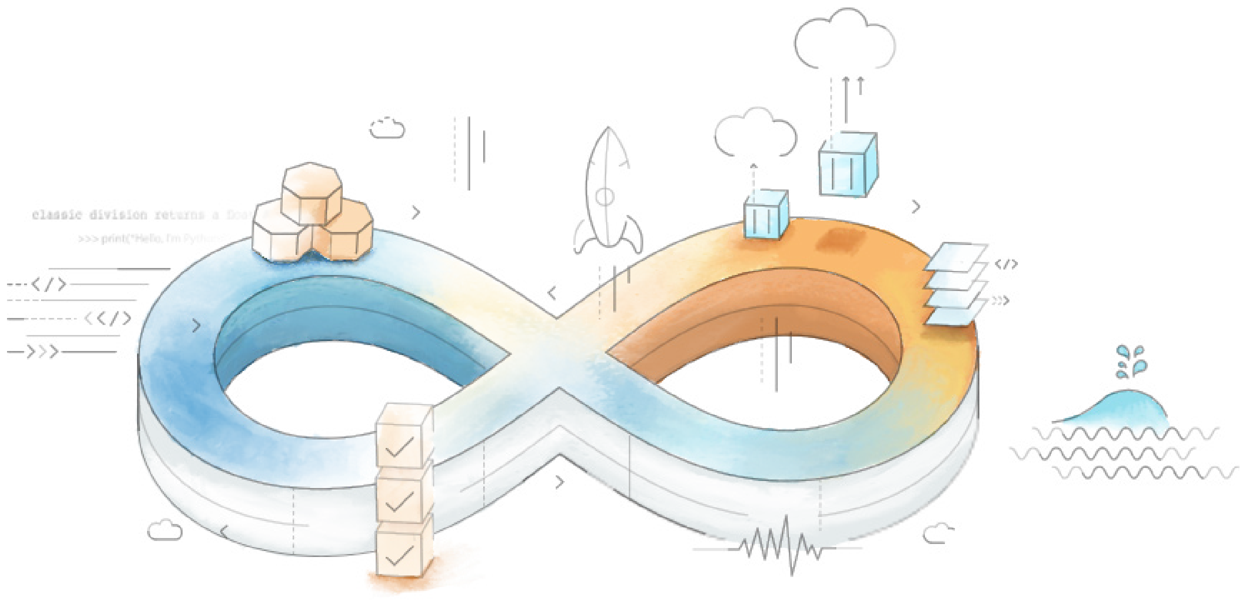
- What is DevSecOps?
- Nine key elements of successful DevSecOps implementations
- Real-life DevOps security challenges
- How to make DevSecOps programs effective
- Four DevSecOps case studies

Table of Contents

DevSecOps: Making It Happen

After understanding the value of a DevOps mindset, making the cultural shift and reaping the benefits, many companies are aiming for the next big step: integrating development, operations, and security into one organization.

However, DevSecOps programs are ambitious and challenging to carry out. Organizations have to manage their resources, tools, cultural and structural changes. This guide explains how DevSecOps programs are implemented in organizations of all sizes and in different verticals, discusses real life challenges faced by organizations making the transition, and provides stories and lessons you can learn from those organizations. We hope this will help you take a big step towards successful implementation of DevSecOps in your organization.



What is DevSecOps?

DevSecOps extends the advantages of a DevOps approach, including agility and responsiveness, to IT security. The idea is to integrate security early in the development process and throughout the Software Development Lifecycle (SDLC). This involves fostering a culture of flexibility and ongoing collaboration between development and security teams, and incorporating security protocols into the development process.

To illustrate, here is a typical DevSecOps workflow:

- **Commit** - a developer creates code in a repository and commits their changes
- **Analyze** - Another developer pulls the code from version control and performs static analysis or peer review to identify security issues or other bugs
- **Setup Environment** - If the code passes static analysis, an environment automatically spins up, deployed in containers with configuration managed by Infrastructure as Code tools
- **Test** - The build system executes a test suite, including unit tests, functional tests, UI automation tests, integration tests, and also security tests
- **Deploy** - If all the tests pass, the new version is promoted to a production environment
- **Monitor** - The new production version is monitored to identify security threats
- **Repeat** - Upon identification of a security issue requiring changes to the software, or any other bug or feature request, a developer creates a new version of the code and commits it to repository, restarting the cycle.

DevSecOps and Container Technology

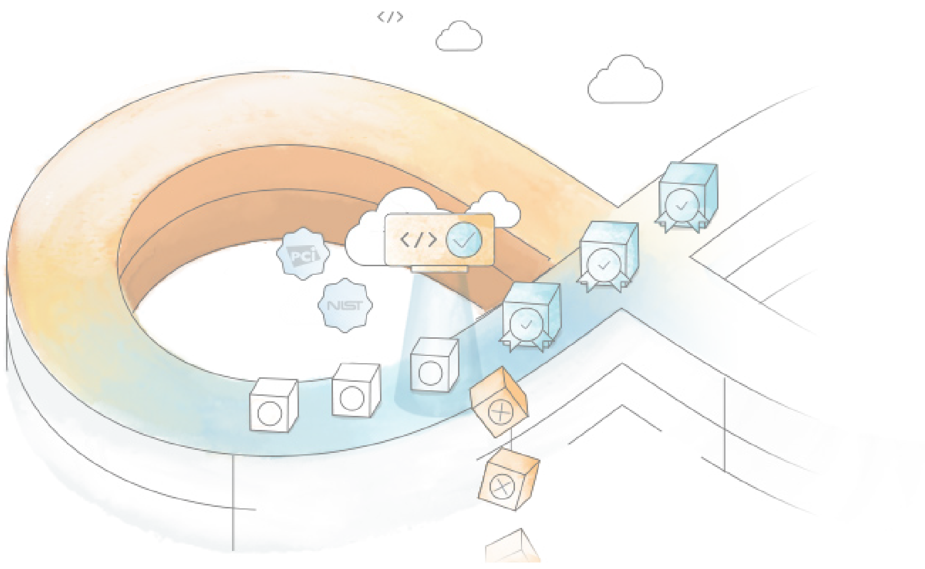
Container technology is an important facilitator of DevSecOps:

- It is predictable and easy to automate, enabling fast development and deployment cycles
- Containers are immutable infrastructure, which makes it easier to identify attacks and recover from them when they are compromised
- Containers make it possible to redeploy updated images frequently - making it easier to remediate security vulnerabilities
- Containers allow teams to shift left, deploying applications automatically and building security testing into the build process, instead of waiting to apply security only after deployment

Other Common DevOps Practices

Other practices common in DevSecOps organizations include:

- Integrating code analysis tools into the development pipeline, to prevent compromised code from reaching production
- Automating and continually testing the production environment to detect weaknesses
- Carrying out fixes and patches before deployment, not after
- Automated or semi-automated security gates that enable careful review of software without slowing down the workflow



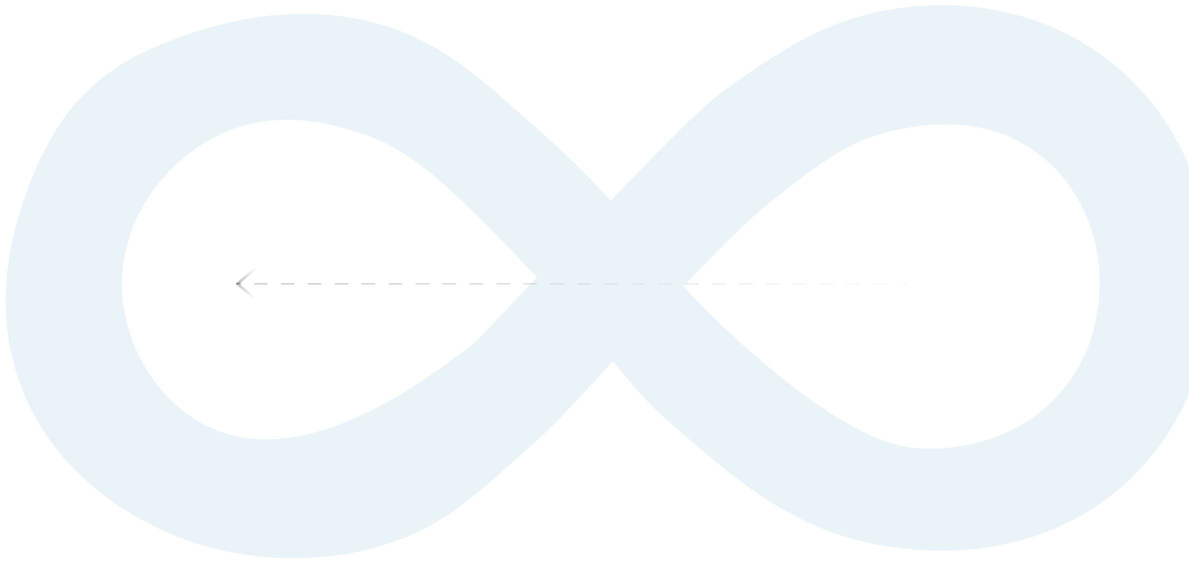
DevSecOps: Shifting Security Left

“Shift left” is an approach to software testing in which tests and fixes are performed earlier in the development pipeline - “more to the left” when looking at a lifecycle diagram that flows from left to right. It enforces the agile principle “test early and often” by building testing into all stages of the SDLC.

In DevSecOps, it is security testing and remediation that shifts left. The core concept of DevSecOps is to address security at all stages of the delivery chain, from requirements and planning to development, testing, deployment and release. The goal is to improve the coverage and effectiveness of security processes, increase software quality, shorten test cycles, and reduce the security debt.

Most importantly - it is easier and less expensive to apply security fixes as early as possible in the cycle. Just like it is exponentially cheaper to fix bugs the earlier you catch them in the development process.

All of the above sounds great on paper, but implementing it is a different matter. In the remainder of this article we’ll dive into **what it takes to really implement DevSecOps**, common challenges organizations run into, and lessons from real-life success stories.



Real-Life DevOps Security Challenges and Overcoming Them

DevOps teams that attempt to take ownership of security and make the transition to DevSecOps, find it is a complex and risky undertaking. Here are a few of the obstacles standing in the way of many DevSecOps transitions and what you can do about them.

The Skills Gap

Most engineers and operations experts aren't experts in security, and security analysts have limited understanding of development processes, tools and tradeoffs. In a DevSecOps organization, complementing each of these groups with the opposite skill has tremendous value. Developers understand security best practices and can start implementing them in every task. Security teams can make more informed suggestions, understanding the implications of specific changes to the software or the environment.

Serverless Security Issues

Serverless architectures, also called Function as a Service (FaaS), allow you to execute and scale business logic without worrying about the runtime environment, storage and operating system. The serverless provider is responsible for the security of cloud infrastructure components, which reduces the security burden placed on the user, but doesn't eliminate it. Your security priorities will need to shift to risks such as unsecured serverless deployment configuration, unsecured storage, over-privileged function permissions and roles, inadequate function monitoring, denial of service (DOS), and financial resource exhaustion by hijacking your own workloads for nefarious purposes (e.g., mining for crypto-currency).

Container Security Risks

Containers facilitate consistent deployment of applications, but they raise some new security challenges. Transient containers and microservices are difficult to monitor, while misconfiguration of container networking can leave your production environment vulnerable. Containers are often used to break down applications into microservices, which increases data traffic. Traditional server security solutions don't support containers; consider specialized security technology that can lock down containers with safe configuration, scan images to ensure they are safe, and monitor containers in production. One such solution is Aqua's Cloud Native Security Platform.

Cloud Deployment Risks

DevOps relies on cloud technology to automate and scale dev/test/production workflows. However the cloud raises new security challenges, including cloud account hijacking, misconfiguration of cloud resources, which can be dangerous if those resources are exposed to the Internet, privileged account abuse which can lead to massive data loss, and insecure or compromised APIs, which are the basis of all cloud integrations.

Legacy Application Security Risks

Security administrators often overlook legacy systems, allowing vulnerabilities to go unchecked. Legacy systems are prone to vulnerabilities and patches must be continuously applied, but in some cases updates might break the legacy system. The presence of legacy applications can disrupt DevSecOps programs; legacy apps can create security threats, but they are difficult to adapt to an automated DevSecOps cycle.

Nine Key Elements of Successful DevSecOps Implementations

Overcoming the key challenges above is only one part of the story. Ensure you address the following best practices in your DevSecOps project:

Integrate automated testing into the pipeline—integrate security testing into the build and deployment process. Test the security of your infrastructure in real-time, or as close to it as possible. To save time, you can automate simple security tests, for example vulnerability scans. You can use automated acceptance tests, known as functional security tests, to verify that features like authentication and logout are working properly. You should choose a testing framework that integrates easily with your CI/CD server, and that your development, security, and ops teams can use comfortably.

Integrate security testing into workflows—there are several types of security testing you can build into your development process. These include code analysis tools like Veracode, which can find vulnerabilities in your code and in open source libraries, cloud security tools like Microsoft Azure Advisor which can advise you on cloud security best practices, and tools for securing container and serverless workloads, alike.

Automated deployment—automation is crucial to continuous delivery workflows. Create a deployment model to manage and orchestrate deployment activities, control variability, and reduce errors. Be faithful to the principle that security should happen before deployment, not after - build security checks and security lock-downs into your deployment process, making it impossible to deploy a non-secured environment.

Infrastructure as Code (IaC)—you can use IaC alongside continuous delivery to manage infrastructure, such as virtual machines and networks. Leverage IaC to control and test installation and deployment processes and ensure they are in line with security practices. Security and ops staff should review and approve configuration flows.

Continuous monitoring—tracking systems in production can contribute to all sides of the DevSecOps triangle. Monitoring can identify security issues and help you resolve them early to prevent incidents in production. Continuous, proactive monitoring helps you improve user experience in production while avoiding expensive rollbacks. Monitoring can also assist development by including integration and acceptance tests to ensure the application works as intended while it is running in production.

Immutable infrastructure provisioning—immutable infrastructure, like containers, are servers that you don't modify after deployment, as opposed to traditional mutable infrastructures in which you continually modify servers. Immutable infrastructure is more consistent, reliable, and predictable, and makes deployment simpler. If you need to fix or update something, you can build new servers to replace the old ones, which are then decommissioned. This mitigates issues like snowflake servers and configuration drift.

Remediating application security vulnerabilities—have clear remediation plans that you can deploy immediately to eliminate vulnerabilities. When using open source software, integrate automated open source management tools to keep you up-to-date with known vulnerabilities and calculate the associated security risk. These tools cross-reference components with an updated database and list vulnerabilities according to priority. Remediate vulnerabilities fast by applying quick, temporary fixes, virtual patching solutions, patching software components, modifying configuration, and using runtime defensive technologies like firewalls or Runtime Application Self-Protection (RASP).

Train engineers in Secure DevOps—Security Awareness Training (SAT) ensures that the development team is familiar with industry standards and can identify, assess, and respond to security issues. Training provides employees with a better understanding of their responsibilities, improves their confidence and performance, and reduces the risk of a breach. A training program can also help foster team cooperation and consistency.

Use secret management tools—secrets embedded in source code and found throughout the DevOps pipeline present a significant security risk. Secrets can be passwords, credentials, tokens, or keys. DevSecOps teams must use tools that securely manage and store secrets. They should also implement access controls that don't affect the automated workflows of DevOps.

Four DevSecOps Case Studies

Allianz, PayPal, the National Institute of Allergy and Infectious Diseases (NIAID), and Maersk are four organizations operating development and operations on a huge scale, who decided to make the transition to DevSecOps. If they could make it happen, anyone can.

Allianz

As a large company, Allianz had trouble ensuring that everyone took responsibility for security and the process for fixing vulnerabilities was slow. Despite having a large in-house IT organization, it was difficult to create automated testing tools and remediation plans. Security was outsourced, making it difficult to upgrade Allianz's infrastructure.

Allianz implemented CloudBees, a continuous delivery platform, to help automate their software development. They also incorporated pen-testing into their DevSecOps process so developers could produce secure code and fix security vulnerabilities quickly. Allianz had to alter the culture of their workplace, which included training their team with a focus on security engineering, threat modeling, and business risk assessment.

PayPal

With over 254 million active account holders conducting billions of payment transactions, PayPal had to find a way to guarantee security at scale. The company planned its transition to DevSecOps with a timeline of less than a year, shifting from a project-driven mindset to a product-aligned approach and prioritizing usability and security.

To help guide the organization through this process, PayPal assigned "Change Champions" and "Transformation Team Members". They created actionable security stories and replaced security lingo with language the development team could understand. They gave autonomy to developers to implement approved security controls and established patterns, providing secure code snippets and offering clear usage guidelines.

PayPal developers create 1 million builds per month, which would be virtually impossible without automated security scans and the flexibility afforded by the DevSecOps methodology.

National Institute of Allergy and Infectious Diseases

The National Institute of Allergy and Infectious Diseases (NIAID) conducts medical research and uses sensitive health data that must be protected. NIAID faced challenges regarding the consistency and predictability of their data, and they found it difficult to incorporate specific, consistent security policies into a systematic framework. However, their first priority was a cultural change.

NIAID updated their security protocols and implemented infrastructure-as-code (IaC) practices, which make application and server configurations inspectable and help reduce vulnerabilities. NIAID also integrated Fortify, a security scanning tool, into their pipeline to prevent the introduction of coding vulnerabilities.

MAERSK

After a cyber attack infected their network in 2017, Maersk had to rebuild their core IT capability in a matter of weeks. This involved reconstructing their server and network infrastructure, updating their global operating system, and restoring their entire application stack. To accomplish this, the shipping company focused on reorganizing and standardizing its digital environment, implementing a new common standard for over 60,000 devices.

Maersk made risk governance a CISO function instead of a central corporate function, so the CISO is responsible for identifying infrastructure gaps and making and enforcing security policies. The CISO makes decisions in consultation with business owners, who can then decide how to address geographically limited risks.

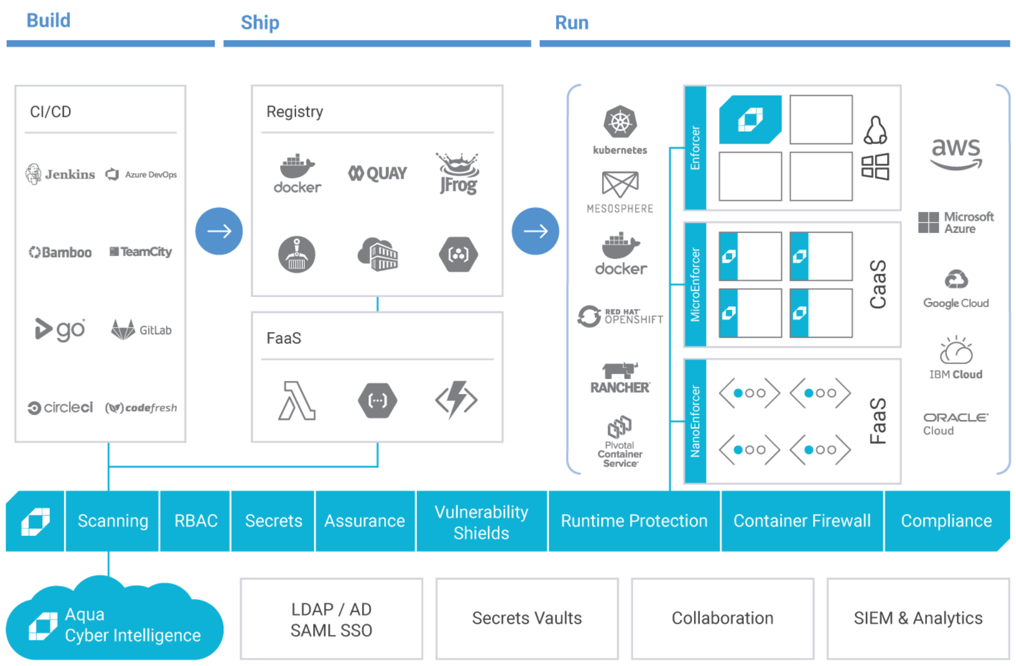
The supervisory board includes non-technical personnel. The company provides readable assessments of its security profile. For example, a funnel diagram represents security data in order of ascending importance, from the number of external surface attacks, through penetrations and security incidents, to major incidents.

DevSecOps for Cloud Native with Aqua Security

Aqua Security lets enterprises secure their cloud-native and container-based applications from development through to production. This is helping bridge the gap between DevOps and IT security and is driving container adoption.

Aqua's Container Security Platform offers transparent, automated security, including full visibility into container activity. Organizations can use this platform to find and prevent malicious activity and attacks. Aqua's platform also helps organizations implement security policies and simplify regulatory compliance.

Organizations can use Aqua to automate the secure development and deployment of applications in their DevSecOps pipelines. With Aqua, organizations can embed comprehensive security testing and effective policy-driven controls early on in the development cycle. The process is fully automated, supporting a shift left strategy.



Aqua helps accelerate application delivery and remove obstacles to your DevSecOps program:

- **Image and function scanning**—scans container images and serverless functions for embedded secrets, known vulnerabilities and malware, licensing issues and configuration problems.
- **Continuous image assurance**—stops unapproved functions and images from being deployed in your environment. It prevents image sprawl and rogue deployments, and preempts operational errors.
- **CI/CD integration**—natively integrates with Azure DevOps, Jenkins, Bamboo, TeamCity, GitLab, and most other tools in the DevOps toolset. Aqua scans images as they are developed and gives actionable feedback to developers within their IDEs.
- **Collaborative remediation**—provides real-time actionable information of configuration remediation and vulnerability, giving feedback to developers within their CI/CD tools, as a ticket in Jira or sent via Slack.
- **Centralized secrets management**—leverages your existing secrets vaults to securely deliver, rotate, and update secrets in containers. This is done with no exposure outside the container and no container restarts.

Find out more about Aqua's Cloud Native Security Platform.