

DevSecOps: Making It Happen



Table of contents

03

DevSecOps:
Making It Happen

09

DevSecOps:
Shifting Security Left

17

Four DevSecOps
Case Studies

04

What is
DevSecOps?

11

Real-Life DevOps
Security Challenges
and Overcoming Them

22

DevSecOps for
Cloud Native with
Aqua Security

06

DevSecOps and
Container Technology

14

Nine Key Elements of
Successful DevSecOps
Implementations

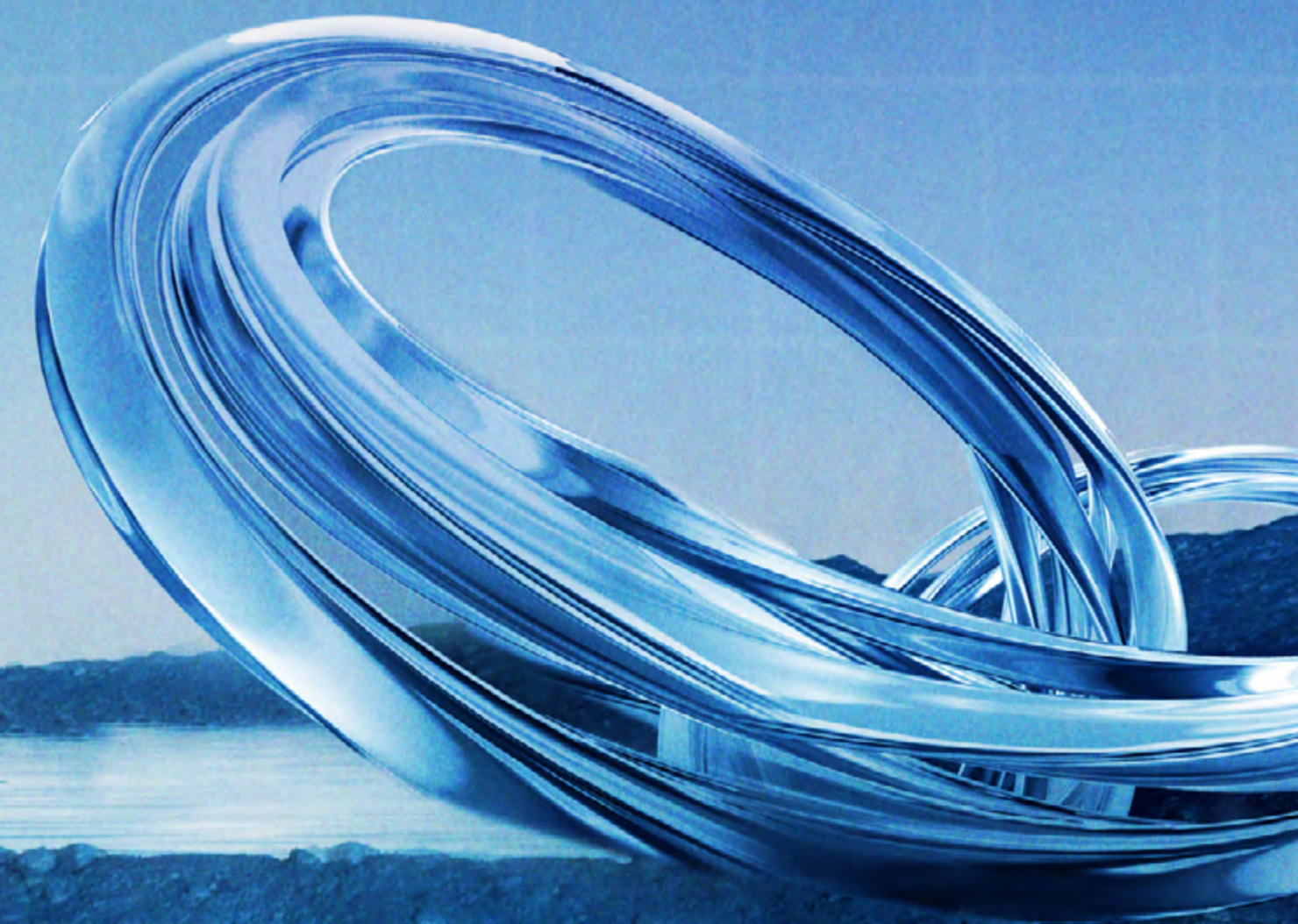
DevSecOps: Making It Happen

DevOps has powered digital transformations across the globe, enabling enterprises to thrive in a rapidly evolving world. This adoption of DevOps has brought highly desired speed to the development process, with a culture of continuous integration and delivery, allowing organizations to release applications faster and more frequently. Digital transformation often involves scaling operations to meet growing demands, which is possible despite the lack of talent with heavily automated DevOps processes that scale the development and deployment processes while minimizing the risk of manual errors.

Yet, in the midst of digital transformation, security took a back seat. With new attacks highlighting the vulnerability of the expanded attack surface, organizations are now embracing DevSecOps. However, executing DevSecOps programs proves to be both ambitious and challenging. Organizations must navigate the intricacies of managing resources, tools, and the nuances of 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.

DevSecOps prevents the agile and automated process of DevOps from being weaponized by an attacker by designing security validation throughout the entire process.



What is DevSecOps?

DevSecOps prevents the agile and automated process of DevOps from being weaponized by an attacker by designing security validation throughout the entire process. The idea is to integrate security early in the development process and throughout the software development life cycle (SDLC). This involves fostering a culture of flexibility and ongoing collaboration between development and security teams and incorporating security protocols into the development process.



A typical DevSecOps workflow:



Software Composition Analysis

The code undergoes software composition analysis to identify and assess open-source components and dependencies, ensuring they meet security standards and compliance requirements.

SBOM Automation

Automated processes generate a software bill of materials (SBOM) that provides a comprehensive inventory of all components, including open-source libraries, used in the code. This SBOM then becomes a critical reference for tracking and managing software components.

Environment Configuration (Infrastructure as Code)

If the code passes static analysis, an automated process spins up an environment. This environment is deployed in containers, and its configuration is managed by infrastructure-as-Code-tools to ensure consistency.

Static Analysis, Peer Review

Another developer performs static analysis or conducts a peer review to identify security issues and other bugs in the committed code.

Commit

A developer creates and commits code changes to version-controlled repository.

Comprehensive Testing

The build system executes a comprehensive test suite, including unit tests, functional tests, UI automation tests, integration tests, and security tests for tracking and managing software components.

Deployment to Production

If all tests pass successfully, the new code version is promoted to a production environment.

Repeat

The cycle repeats as needed, fostering a culture of continuous improvement and adaptation to evolving security and development needs.

Learning and Iteration

The team conducts a retrospective to learn from the cycle. Insights gained are used to iterate and improve the DevSecOps workflow continually.

Documentation and Communication

Documentation is updated to reflect changes, and relevant stakeholders are informed about the updates.

Automated Remediation

Automated processes kick in to remediate the identified security issue or bug, ensuring a swift response.

Incident Response

In the event of a security issue or any other bug or feature request identified during monitoring, a developer creates a new version of the code and commits it to the repository.

Continuous Monitoring

The production version undergoes continuous monitoring to promptly identify and respond to any security threats.

DevSecOps and Container Technology

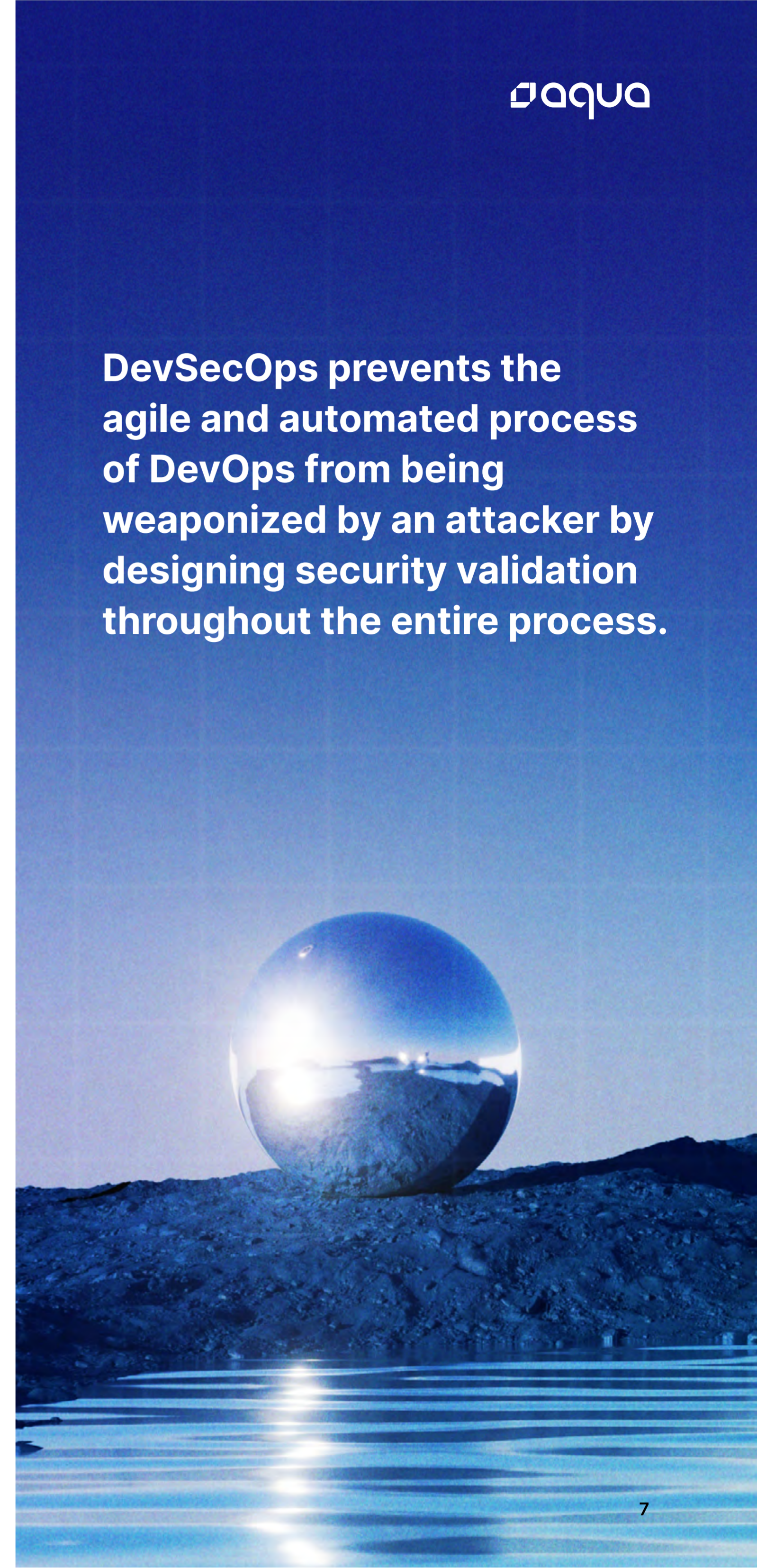
DevOps has powered digital transformations across the globe, enabling enterprises to thrive in a rapidly evolving digital world. This adoption of DevOps has brought highly desired speed to the development process, with a culture of continuous integration and delivery, allowing organizations to release faster and more frequently.



Containers offer predictability and ease of automation, fostering faster development and deployment cycles (a key ingredient of a mature DevSecOps program). The packaging of applications and their dependencies ensures consistent behavior across various environments, streamlining the development process and allowing for better automation of deployment pipelines.

Secondly, containers enable the concept of immutable infrastructure, providing a substantial advantage in security. The unalterable nature of containers simplifies the identification of potential security breaches. In the event of a compromise, the immutable state allows for efficient recovery by isolating the compromised container and deploying a new, clean instance. This modern resilience minimizes the impact of security incidents and enhances the overall security posture of the system.

DevSecOps prevents the agile and automated process of DevOps from being weaponized by an attacker by designing security validation throughout the entire process.



Securing containers in a DevSecOps environment introduces another layer of risk that poses unique challenges:

Vulnerability Management

Containers often rely on various dependencies and base images. Managing and keeping track of vulnerabilities in these components can be challenging, especially as containerized applications evolve rapidly.

Container Orchestration Security

Orchestration tools (ie. Kubernetes) introduce additional opportunities for misconfigurations, insecure defaults, or inadequate access controls.

Runtime Security

Once containers are deployed, monitoring and securing them in real time is crucial. Runtime security challenges include detecting unauthorized access, preventing lateral movement, and addressing potential threats during execution.

Image Security

Ensuring that only approved and secure images are used in the development and deployment process requires continuous monitoring and validation.

Data Management and Regulatory Compliance

Containers often need access to data, and managing sensitive data securely within a containerized environment can be challenging. Encryption, access controls, and data lifecycle management need careful consideration.

DevSecOps: Shifting Security Left



Here are a few of the obstacles standing in the way of many DevSecOps transitions and what you can do about them

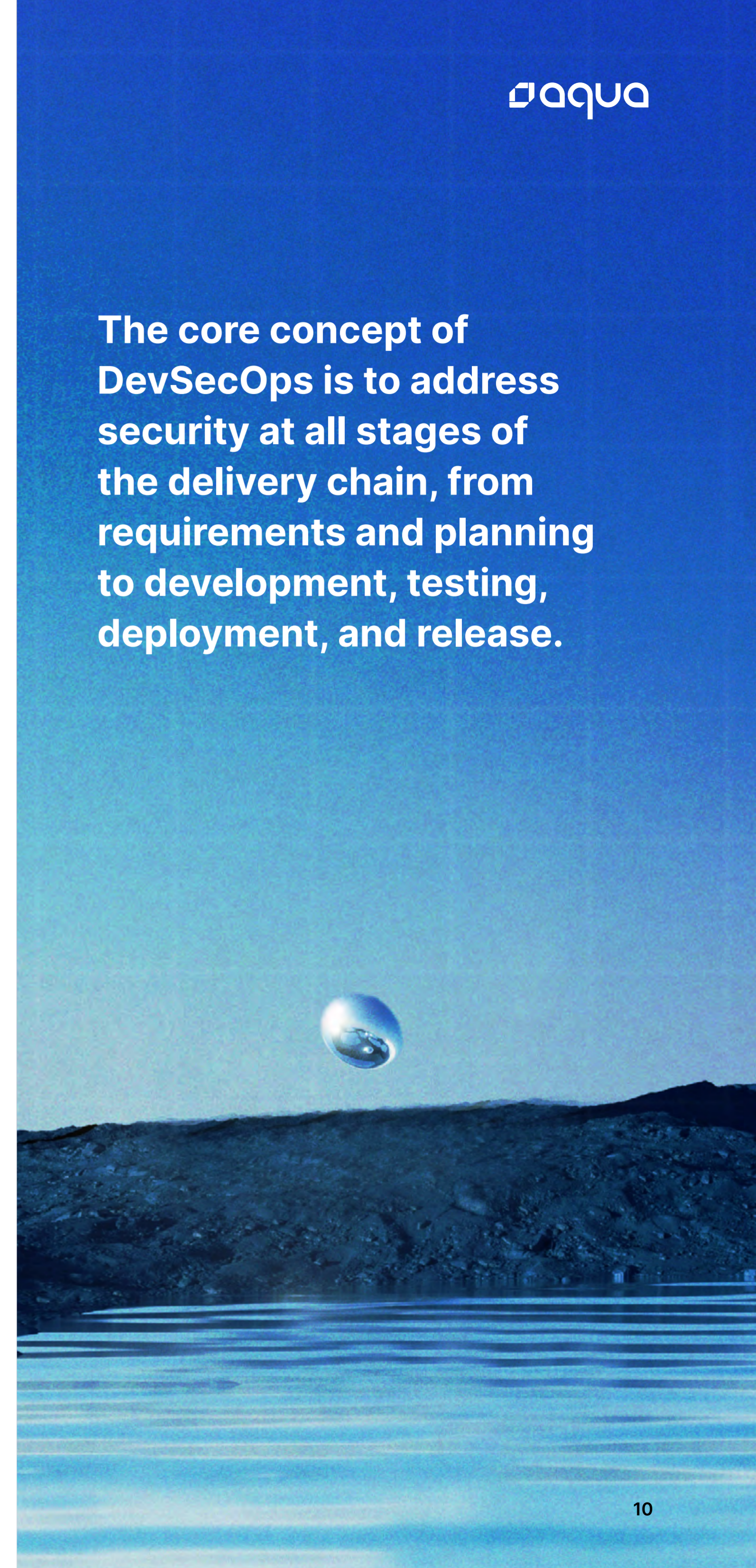
“Shift left” is an approach to software testing in which tests and fixes are performed earlier in the development pipeline — “more to the left” when looking at a lifecycle diagram that flows from left to right. It enforces the agile principle “test early and often” by building testing into all stages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

In DevSecOps, it is security testing and remediation that shift 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's exponentially cheaper to fix bugs the earlier you catch them in the development process.

All 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.

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.



Real-Life DevOps Security Challenges and Overcoming Them

DevOps teams that attempt to take ownership of security and make the transition to DevSecOps find that it's 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.

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.

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 the hijacking of your workloads for nefarious purposes (e.g., mining for crypto currency).

Cloud Deployment Risks

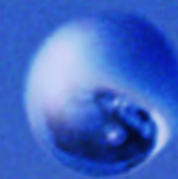
DevOps relies on cloud technology to automate and scale dev/test/production workflows. However, the cloud raises new security challenges, including cloud account hijacking; misconfiguration of cloud resources, which can be dangerous if those resources are exposed to the Internet; privileged account abuse 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.

9 Key Elements of Successful DevSecOps Implementations

Overcoming the key challenges above is only one part of the story.



You also need to ensure you address the following best practices in your DevSecOps project:

1

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.

2

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 Trivy (Aqua's open source scanner), 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.

3

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 in a non-secured environment.

4

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.

5 **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.

6 **Immutable infrastructure provisioning** - Immutable infrastructures, 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.

7 **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 and virtual patching solutions, patching software components, modifying configuration, and using runtime defensive technologies like firewalls or Runtime Application Self-Protection (RASP).

8 **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.

9 **Use secrets 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.

4 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.





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.



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.



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 cultural change.

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



After a cyberattack infected their network, 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.

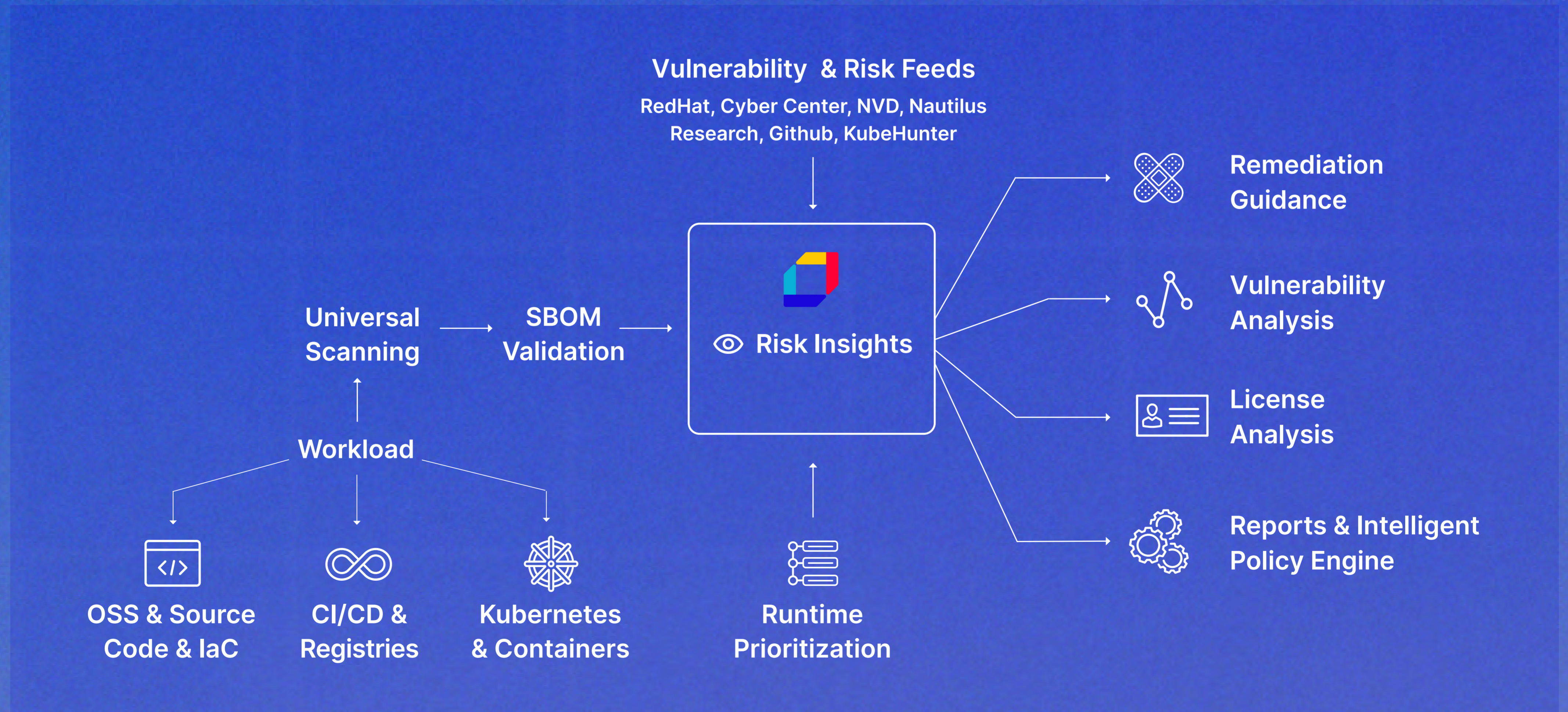
DevSecOps for Cloud Native with Aqua Security

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.



Container security with the Aqua Platform offers transparent, real time 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 in the development cycle. The process is fully automated, supporting a shift left strategy.



The Aqua Cloud Native Application Protection Platform (CNAPP) plays a pivotal role in accelerating a DevSecOps approach by integrating a comprehensive set of capabilities that enhance security throughout the entire application lifecycle.

Here's a breakdown of how Aqua CNAPP achieves this:

1

Software Bill of Materials (SBOM) - Aqua CNAPP incorporates SBOM capabilities to generate and maintain a detailed inventory of all software components and dependencies used within an application. This provides transparency into the software supply chain, aiding in tracking and managing potential vulnerabilities.

2

Software Composition Analysis (SCA) - The platform includes robust SCA features, allowing organizations to scrutinize open source components and third-party libraries for known vulnerabilities. Aqua CNAPP ensures that only secure and compliant components are integrated into the application, reducing the risk of exploiting vulnerabilities.

3

Static Application Security Testing (SAST) - Aqua CNAPP integrates SAST into the development pipeline, enabling the early detection of security flaws in the source code. By automating SAST processes, developers receive prompt feedback on potential vulnerabilities during the development phase, facilitating quicker remediation efforts.

4

Container Security - The Aqua platform provides specialized container security capabilities, addressing vulnerabilities and misconfigurations specific to containerized environments. It ensures that containers adhere to security best practices, minimizing the risk of container-based threats and attacks.

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

Aqua streamlines and fortifies the DevSecOps approach by embedding security practices throughout the entire software development life cycle.

5

Runtime Security - Real-time protection is a core feature of the Aqua platform. By continuously monitoring containerized applications during runtime, the platform detects and mitigates potential security threats, unauthorized access, and anomalous behaviors. This dynamic approach enhances security post-deployment.

6

Centralized secrets management - leverage 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.

7


CI/CD Integration - Aqua CNAPP seamlessly integrates with CI/CD pipelines, embedding security checks directly into the development process. This ensures that security assessments, including vulnerability scans and policy enforcement, are an integral part of the automated deployment pipeline, promoting a shift-left approach.

8

Image Scanning - Aqua CNAPP conducts thorough image scanning to identify and remediate vulnerabilities within container images. This pre-deployment scanning ensures that only secure and compliant images are deployed into production, preventing potential security risks from being introduced.

9

Threat Analysis Across the Application Lifecycle - The Aqua platform is unique in its holistic threat analysis across the entire application lifecycle. By examining threats comprehensively from development through runtime, it prioritizes true threats based on their impact and severity. This helps developers focus their remediation efforts on addressing the most critical security issues first.



Aqua CNAPP streamlines and fortifies the DevSecOps approach by embedding security practices throughout the entire software development lifecycle. From early code development to continuous monitoring in runtime, Aqua CNAPP provides a unified platform for threat analysis, vulnerability management, and remediation efforts, ultimately enhancing the security posture of cloud-native applications.

Aqua Security sees and stops attacks across the entire cloud native application lifecycle in a single, integrated platform. From software supply chain security for developers to cloud security and runtime protection for security teams, Aqua helps customers reduce risk while building the future of their businesses. The Aqua Platform is the industry's most comprehensive Cloud Native Application Protection Platform (CNAPP). Founded in 2015, Aqua is headquartered in Boston, MA and Ramat Gan, IL with Fortune 1000 customers in over 40 countries. For more information, visit <https://www.aquasec.com>



[Schedule demo ›](#)