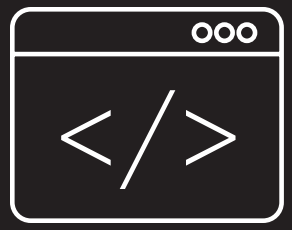
PRINCIPLES FOR
Securing DevOps

VERACODE

INTRODUCTION

The DevOps Difference

DevOps, a new model for software development, is transforming the way the world creates software. And despite its substantial organizational, cultural and technological requirements, this new way of organizing development and IT operations work is spreading rapidly.



THE RESULT

better software and a better bottom line.

A recent study found that companies that have embraced DevSecOps enjoy a significant competitive advantage.[1] Study results show that these companies are:

- **2.6 times** more likely to have security testing that can keep up with frequent app updates
- **2.4 times** more likely to be leveraging security to enable new business opportunities
- **2.5 times** more likely to be outpacing their competitors

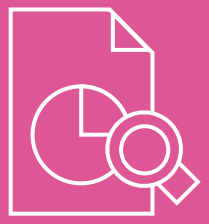
They also have a 50% higher profit growth and 40% higher revenue growth.

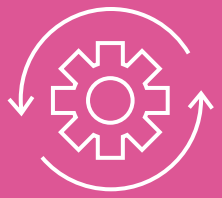
[1] *Integrating Security Into the DNA of Your Software Lifecycle*, Freeform Dynamics, January 2018.

In addition, continuous delivery disciplines, like test and deployment automation, version control of app and system configuration, trunk-based development and continuous integration, all lead to better IT performance and higher organizational performance.

But these new gains require a new way of thinking about application security. To succeed, it's critical to understand how DevOps and continuous integration/continuous deployment (CI/CD) are different from Agile development and how this difference changes the requirements for your application security.

THIS PAPER



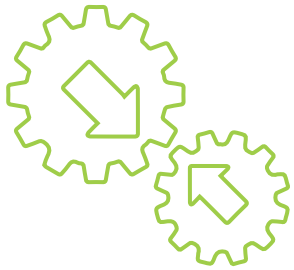


Provides background on the continuing evolution of DevOps

Proposes five requirements for integrating application security with DevOps

DevOps Evolution Meets Revolution

DevOps enables software development teams to more consistently reach their goals for the on-time delivery of high quality software by removing organizational barriers between Agile development teams and non-Agile supporting processes.



The fact is, many Agile software projects have succeeded in improving their quality practices, only to face failed deployments when unanticipated operational requirements resulted in software that didn't meet the needs of availability, scalability or manageability.

By integrating activities and organizations, like operations, earlier in the development process, DevOps exposes the development team to these potentially surprising or disruptive requirements early, so they can plan for and address them ahead of time.

As DevOps thought leader Gene Kim has noted, DevOps practices explicitly seek to align the potentially at-odds goals of "make changes quickly" (development) with "keep everything stable" (IT operations). It does this by bringing the teams together to share responsibility for software delivery and operation. This organizational alignment supports all the other activities of DevOps.

In this way, DevOps is a natural evolution of Agile software development and its culture of "retrospectives," "do better" and clearing blockages to getting work done. The revolutionary difference is clearly seen in how — and how frequently — the software is delivered to market.

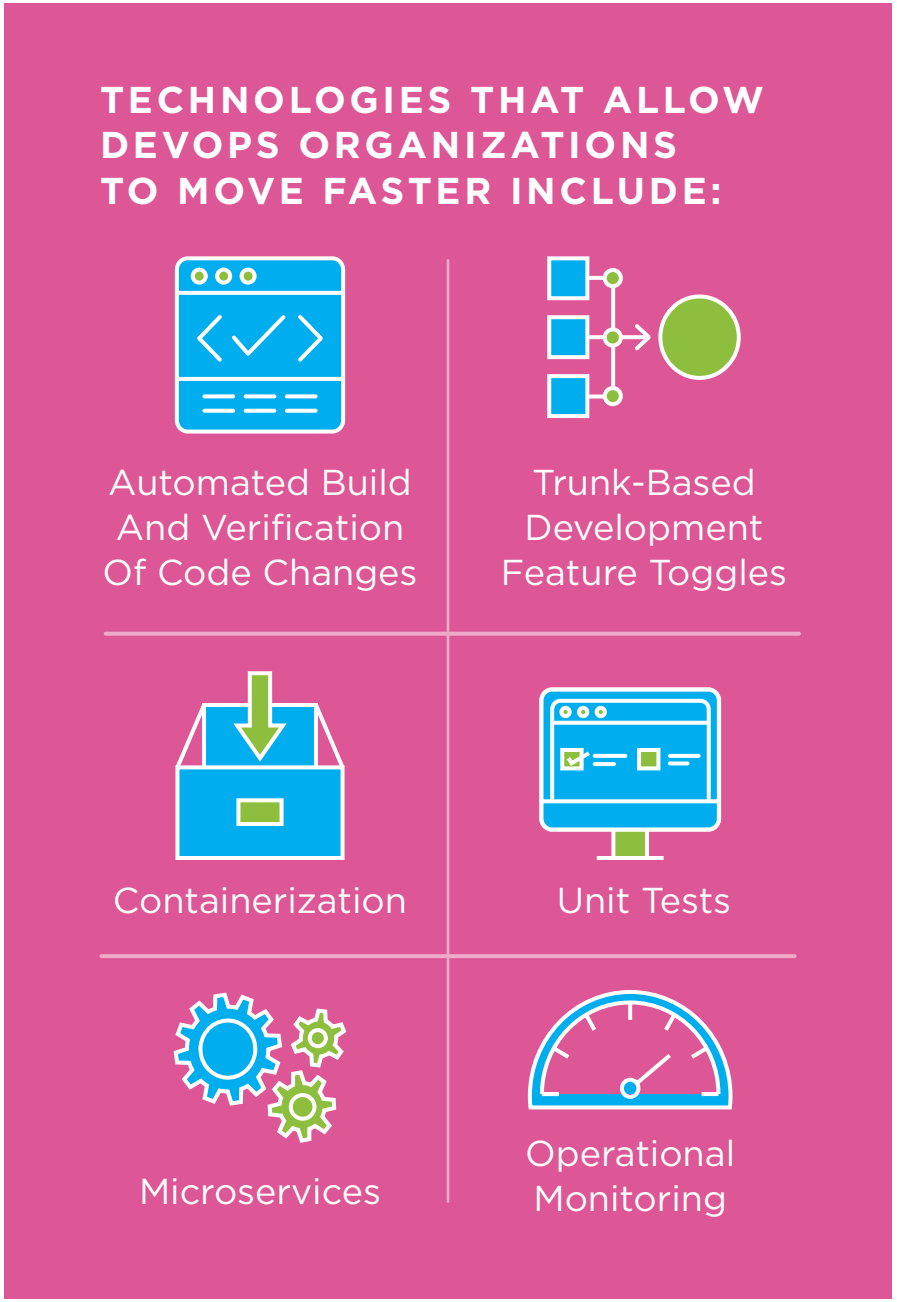
FOR INSTANCE, DEVOPS:

1. Embraces continuous integration and its transformation into continuous deployment

2. Implements insights from the traditional manufacturing quality control process to the software development process

If DevOps includes cultural, organizational and technological components, its sister technologies — CI/CD — are the technological foundation on which DevOps builds its practices.

CI/CD automates much of the routine work of transforming code changes into working software, including delivering tested code into production. From its roots in build servers like Hudson, Jenkins and Microsoft Team Foundation Server, CI/CD has become a collection of technologies and practices that support the integrated mission of releasing new code changes while keeping everything stable.



"Shifting Security Left" Drives New Requirements for AppSec

Like operations, security's goal of minimizing enterprise risk can sometimes seem to be at odds with development's mandate for rapid change. In reality, there's a middle path that allows development to deliver more secure code at DevOps speed, but it requires security to meet development halfway by adapting the following five DevOps principles:

1
Automate Security In

2
Build Security Champions

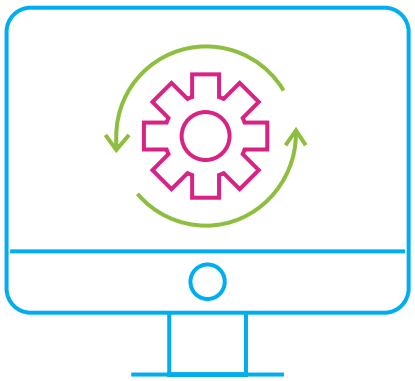
3
Integrate to "Fail Quickly"

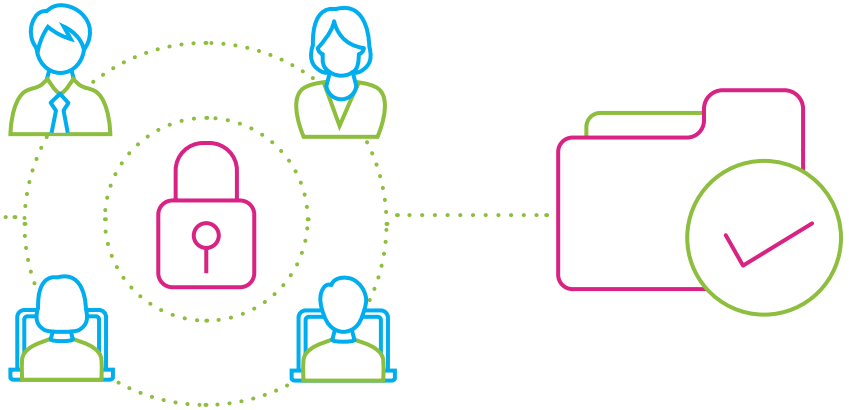
4
Keep Operational Visibility

5
No False Alarms

PRINCIPLE ONE

Automate Security In



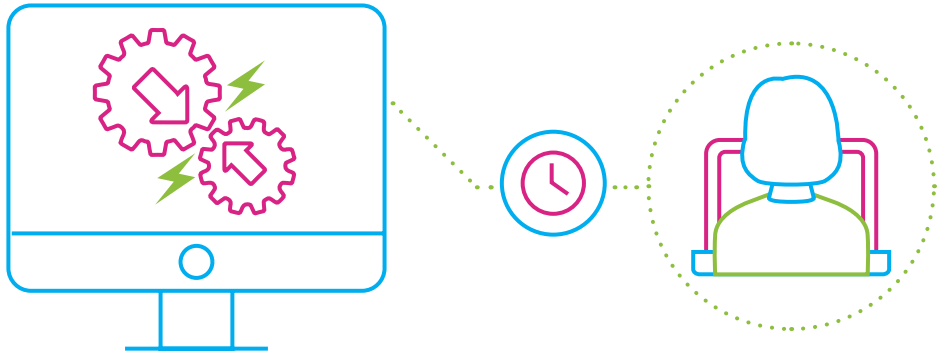


The first step in the DevOps process is to automate security from day one. It not only makes the whole process repeatable and faster, but it fits with the way development teams work. By automating security, teams can ensure that each component gets all the security testing it needs without taking up any extra resources, making security a part of the development process itself.

The key to automating security is to look at the tools you already have in place. Identify the ones you use for continuous integration, continuous deployment, defect tracking and more to determine the best points to introduce automation. The more you can automate security to make it a part of the developers' process, the less work a separate security team will need to do later on.

PRINCIPLE TWO

Integrate to "Fail Quickly"



Next, integrate security into the CI/CD process as early in the development lifecycle as possible. Your goal should be to minimize the gap between the discovery of a problem and the time it takes to bring the developer back in to fix it.

That's because it's much faster, cheaper and easier to ask a developer to fix something they just coded compared to something they wrote six months ago. The longer you wait, the longer it will take for the developer to get back up to speed with that particular code, assuming the developer is even still on the project. Aim to conduct security testing during continuous integration. Better yet, use automation to conduct testing while the developer is writing the code, giving them instant feedback so they can make a fix before it ever becomes a problem.

However, that doesn't mean security is now the responsibility of developers. It would be unrealistic to expect developers to have the same expert understanding of the threat space their applications will face in the same way AppSec professionals do. In addition, developers are often motivated to deliver code as quickly as possible, creating a potential conflict between fixing a security issue now versus putting it off until later. AppSec must be a partnership between development and security, where security defines the acceptable security quality level and developers implement continuous testing to address issues as they appear.

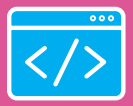
When it comes to deciding when to perform security testing during development, several factors will come into play:

SHIFT SECURITY LEFT

Working with web applications like microservices that make it simple to test each component individually

Using modern development languages, which can support tools that perform security testing directly in the developer's IDE while the developer codes

SHIFT SECURITY RIGHT

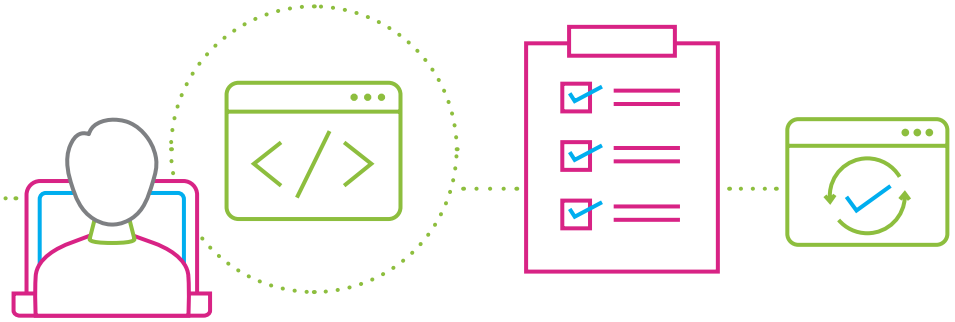


Working with legacy applications that may be more monolithic

Working with languages that automated security testing tools don't support

PRINCIPLE THREE

No False Alarms

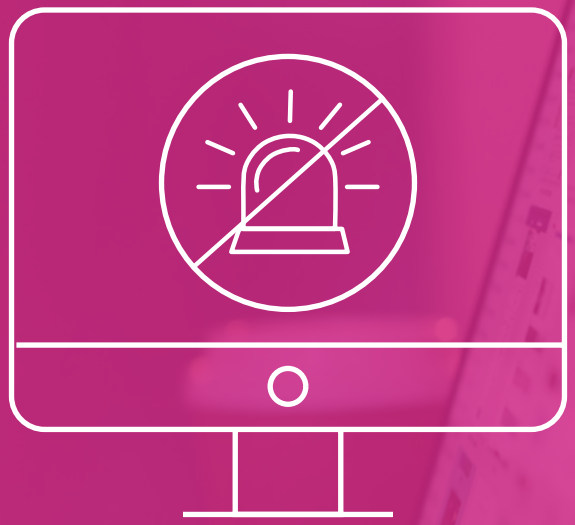


Any test that creates too many false positives is as bad as one that lets flaws slip through the cracks. This is especially true in CI/CD, where a failed security test can stop a critical business function from being delivered to production, or a critical patch from being released. That's exactly what should happen if the security issue is real, but it's completely unacceptable if the finding is a false positive.

Earlier generations of AppSec tools often returned their fair share of false alarms due to the way they were designed. These tools were intended to be used by an AppSec professional to show all the potential issues a piece of software might have; the AppSec pro would then screen them before passing the actual issues on to the developer.

However, that approach falls apart when the developer is getting results directly without AppSec's involvement. If your automated testing has too many false positives, you'll unnecessarily stop the development team's delivery process, force them to deal with the false positive findings, delay their schedule and put the delivery of business value the software represents at risk.

Instead, modern tools are designed to provide both maximum coverage for finding critical flaws while tuning out the noise of low-level issues. That way, developers aren't bothered by false alarms that make them ignore the process or remove it from their workflow.



PRINCIPLE FOUR

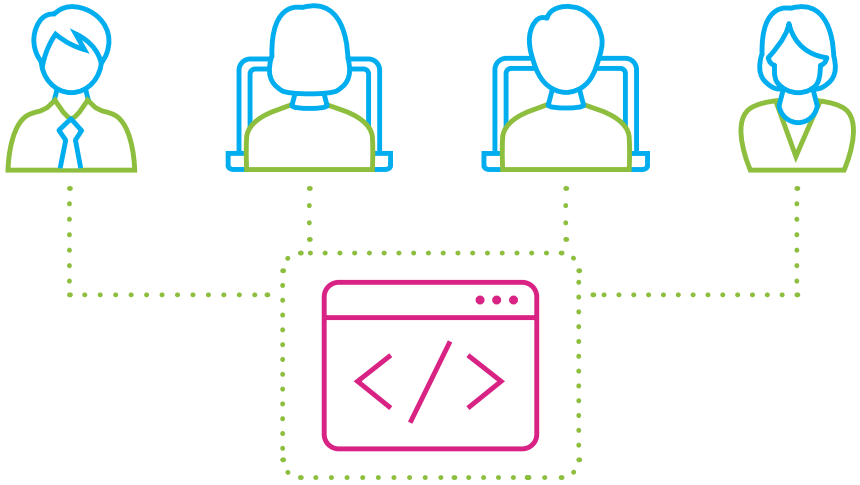
Build Security Champions

Most developers aren't trained in the practices of secure coding. By embedding application security knowledge directly in the team, you can give your security team a force multiplier while reducing culture conflict.

Start by designating one person on the development scrum team as the go-to resource for application security. When you have a security champion embedded with every scrum team, you're guaranteed to have security represented in every design decision and at every stand-up.

This helps remove the security bottleneck that can occur when you have a traditional centralized AppSec expert who's outside of development; it also helps the security team be represented in more places at once.

In a DevOps environment, security isn't responsible for making sure an application is secure — everyone is. Security's job then becomes more about giving the development team the tools, process, expertise and governance needed to empower them to find and fix flaws in their code.



HERE ARE A FEW WAYS YOU CAN CREATE A CULTURE OF SECURITY:

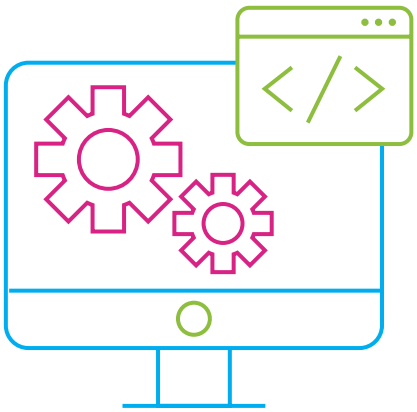
Give your security champions a chance to regularly get together with your security experts to discuss security issues in the news and build a community focused on security within the company.

Make sure security starts at the top by getting engineering management and leadership to prioritize security as a business metric.

Help security champions further their thinking with regular training from your security experts so they can take those best practices back to the rest of the development team.

PRINCIPLE FIVE

Keep Operational Visibility



In a DevOps approach to software development, the development team's job doesn't stop once the product is in production. Instead, the team is responsible for the product throughout the process, making the team responsible for dealing with any security issues that come up in the production environment.

Development teams should also continue to monitor security in live-running applications. Once they can understand how applications are being attacked in the real world, they can take corrective action faster and more effectively. This can be done with a variety of tools and practices, many of which the team already uses, though their tools may need tuning to isolate attacks from other events.

Because one vulnerable application can pose a risk to the entire organization, teams must understand the entire attack perimeter of their organization to ensure all applications have been subjected to the right degree of security scrutiny in the development process. All too often, applications are only considered in a silo based on the line of business using them, without considering the organization as a whole.

Ask yourself these questions when designing an integrated solution for securing the CI/CD pipeline:

- → Have you rearchitected your applications for microservices, or is that work still in progress?
- → Which of your applications will pass through a CI/CD pipeline? Microservice-based? Monoliths? In what languages?
- → What tolerance do you have for "false alarms" from an application security capability that's integrated into your DevOps practices?
- → Are you practicing trunk-based development, or do you still practice release-and-feature branching?
- → How do you plan to monitor your operational applications for security attacks?
- → How do you plan to bring security expertise into the DevOps team?

The process and technical requirements for integrating security with DevOps practices and CI/CD technology can be a challenge. But it's a challenge that can have highly rewarding results.

If you're ready to get serious about securing your own DevOps process, learn how Veracode integrations make this as easy as it gets.

→ **WATCH THE WEBINAR**

"Bringing Security to DevOps with Veracode Integrations."

VERACODE

Veracode is a leader in helping organizations secure the software that powers their world. Veracode's SaaS platform and integrated solutions help security teams and software developers find and fix security-related defects at all points in the software development lifecycle, before they can be exploited by hackers. Our complete set of offerings helps customers reduce the risk of data breaches, increase the speed of secure software delivery, meet compliance requirements, and cost effectively secure their software assets — whether that's software they make, buy, or sell.

Veracode serves more than 1,400 customers across a wide range of industries, including nearly one-third of the Fortune 100, three of the top four U.S. commercial banks, and more than 20 of Forbes' 100 Most Valuable Brands. Learn more at **www.veracode.com**, on the **Veracode blog**, on **Twitter** and in the **Veracode Community**.