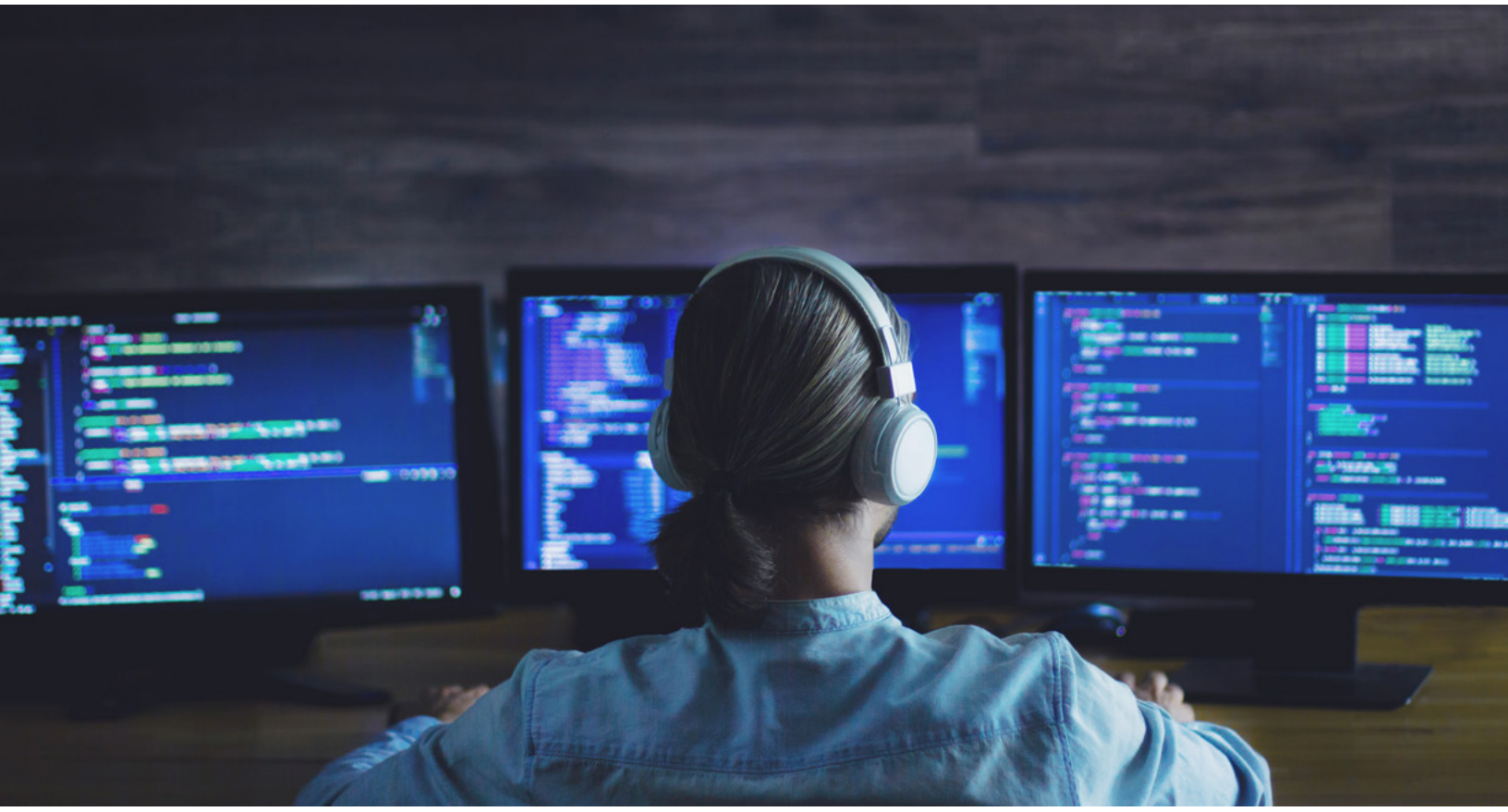
McKinsey Digital

# Agile, reliable, secure, compliant IT: Fulfilling the promise of DevSecOps

By integrating security into DevOps, companies can step up the speed and frequency of software releases without compromising controls or increasing risk.

*by Santiago Comella-Dorda, James Kaplan, Ling Lau, and Nick McNamara*





May 2020

**As digital technologies** transform industry after industry, businesses are increasingly adopting modern software-engineering practices pioneered by tech companies. Agile, DevOps, and other methods enable organizations to test, refine, and release new products and functionality more rapidly and frequently than ever before. However, the speed and frequency of releases can come into conflict with established methods of handling security and compliance. How can organizations resolve this tension?

We think the answer lies in DevSecOps, a method for integrating security into agile and DevOps efforts across the whole product life cycle. Properly implemented, DevSecOps (literally, Development, Security, Operations) offers enormous advantages.

Companies can increase the frequency of software releases from quarterly to weekly, or even daily, without compromising their risk posture. They can cut mean time to remediate vulnerabilities from weeks or months to hours as well as eliminate delays, cost overruns, product defects, and vulnerabilities. Last, but not least, getting security and compliance right from the outset is imperative as companies' growing dependence on digital technologies makes them more vulnerable to cyberattack, especially in the wake of the uncertainty and confusion wrought by the coronavirus pandemic.[1]

In our experience, the companies that are most successful at extracting the full value from DevSecOps commit to managing technology differently. They have an integrated operating model made up of teams of people—including those from security and compliance—with the full range of necessary capabilities, make practical use of automation, develop secure modular services that are easy to use, and conceive of and build digital products that are secure by design.

## What is DevSecOps?

Pioneered by digital-native companies, DevSecOps is based on the principle of integrating development, security, infrastructure, and operations at every stage in a product's life cycle, from planning and design to ongoing use and support (exhibit). This enables engineers to tackle security and reliability issues more quickly and effectively, making organizations more agile and their digital products and services more secure and reliable. Security, reliability, and compliance considerations are built into every agile sprint rather than being handled separately or left until the end of the development process.

Adopting a DevSecOps approach has implications for each stage of the product life cycle:

— *Planning.* From the inception of a new product, teams are aware of their security and reliability responsibilities and trained to handle them. For significant efforts, teams start by quickly modeling threats and risks and then identifying and prioritizing backlog[2] items needed to make the product secure, reliable, and compliant. Where possible, teams take advantage of existing architectural designs that have been developed in collaboration with security and reliability experts, thereby ensuring that best practices are observed as well as speeding up planning and design.

— *Coding.* To improve code quality, developers constantly develop and update their knowledge of secure and resilient coding practices. They take full advantage of reusable coding patterns, components, and microservices to quickly build the functionality and services needed to meet common security and resiliency requirements for encryption, authentication, availability, and observability.

[1] See Jim Boehm, James Kaplan, and Nathan Sportsman, "Cybersecurity's dual mission during the coronavirus crisis," March 2020, McKinsey.com.
[2] A backlog is a prioritized list of the features that an agile product team is planning to build and work on.

Exhibit

## Security is integrated into every step in the product-development life cycle.

### Best practices in DevSecOps

**Agile teams are aware of their security responsibilities** from the outset; security champions are embedded in teams

**Teams quickly model threats** for all significant efforts

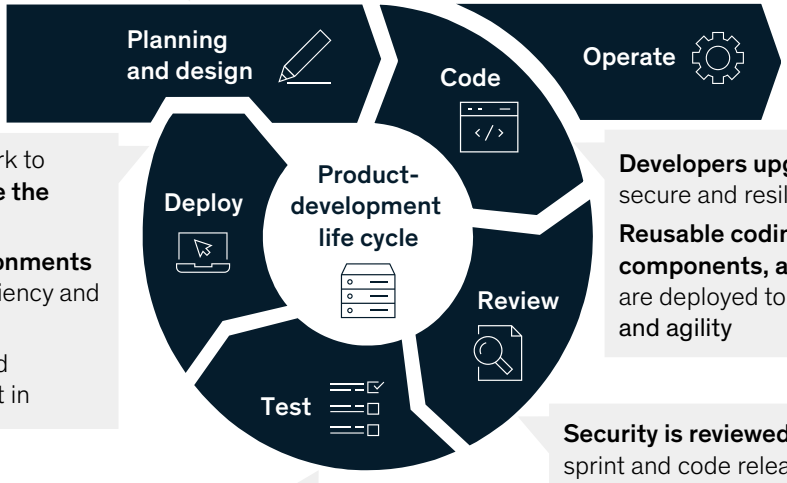
**Backlog items are created, prioritized, and tracked** to meet security and reliability requirements

**Secure architecture designs are preapproved** for implementation

**Real-time monitoring** of app run time ensures potential security issues are identified

**Host and network-based intrusion detection** is implemented

**Compliance validation** and evidence gathering are automated




Engineering teams work to **progressively improve the path to production**

**Secure hosting environments "as code"** ensure efficiency and repeatability

**Strong encryption** and authentication are built in

**Developers upgrade their skills** in secure and resilient coding practices

**Reusable coding patterns, components, and microservices** are deployed to improve security and agility

**Security is reviewed** as part of every sprint and code release

**Automated code analysis tools** (SAST[1]) are used to validate security

**Senior developers with secure coding expertise** conduct peer reviews

**Security test cases are developed and automated** by agile team members

**Automated penetration testing** (including DAST and IAST[2]) is performed as part of the development process

[1] Static application security testing.
[2] Dynamic application security testing and interactive application security testing.

— *Reviewing.* Instead of having a specialist group scrutinize a product for security vulnerabilities and resiliency issues once it emerges from months of development, teams review code as often as every two weeks as part of regular agile sprints, using both automated and manual checks. After automated code-analysis tools such as SonarQube and Fortify have looked for known vulnerabilities and issues, senior developers conduct peer reviews to discuss the results and ensure the software meets appropriate standards.

— *Testing.* Engineers create automated security tests to be run alongside automated functional and performance tests. This not only ensures that testing is consistent and efficient but also makes security requirements explicit, so that developers don't waste time puzzling over how to satisfy ill-defined policies laid down by separate groups. Common security tests, such as penetration tests that look for security holes in systems, are conducted automatically as part of every sprint and release cycle.

— *Deployment.* Code is delivered to production hosting environments, not through manual processes itemized in checklists, but via well-engineered automated processes that ensure the right software is built and that it is deployed securely and reliably. In addition, best-practice companies have secure production hosting environments that can be rapidly invoked through application programming interfaces (APIs), eliminating wait times and reducing risk.

— *Operations.* Once software is in production, automated processes—including real-time monitoring, host- and network-intrusion detection, and compliance validation and evidence attestation—are used to increase efficiency and detect vulnerabilities. If defects or vulnerabilities are discovered, resolutions are identified, prioritized, and tracked to make sure product reliability and security are constantly improved.

## Adopting DevSecOps principles

Capturing the potential of DevSecOps isn't easy. It relies on tight collaboration both within IT and across IT, security, compliance, and risk. To get it right, companies need to make four shifts in the way they manage technology: create a more integrated operating model, build secure "consumable" services, automate development and release processes, and evolve product architectures.

To illustrate what these shifts look like in practice, we'll draw on examples from two organizations that have recently adopted DevSecOps principles: a global software-and-services company and a large financial-services provider. The software-and-services company was already using agile methods to develop digital products and manage infrastructure, but it was striving to improve the resiliency of its products while making its internal processes more efficient. By contrast, the financial-services firm still relied on traditional waterfall methods to deliver its projects, and it saw DevSecOps as a way to improve performance, accelerate software releases, and streamline controls without increasing risk.

### Organization and talent: Integrated cross-functional teams

When organizations struggle with the tensions between being agile and maintaining security, reliability, and compliance, it's often because the skills and accountabilities for developing, operating, and securing products and validating compliance are split between different groups. The answer is to break down these silos by setting up integrated agile teams charged with solving *all* the requirements of the products in their scope, regardless of any functional, security, reliability, or compliance issues they may pose. These teams should be staffed not with specialists but with well-rounded "full-stack" engineers who can work across disciplines and pick up new skills quickly. Every team member must be responsible for the security and reliability of the code they create, whether it's for customer-facing products or internal shared services.

The software-and-services company mentioned earlier reconfigured its product-development teams to own their code bases from end to end instead of relying on separate teams to address maintenance and defects. The company also set up site reliability engineering (SRE) teams to help improve the way products were developed and operated. The two teams worked closely together, shared objectives and key results (OKRs), and took part in joint agile events to stay aligned on

how to improve the security and reliability of the company's products.

The financial-services firm took a slightly different approach, embedding SREs in product-development teams instead of having them in a separate team. But similar to the software-and-services company, it introduced security and reliability OKRs for those product teams as well as common service metrics to drive alignment and accountability.

**Consumable services: Developer-owned operations**

In the past, the accountability for a digital product's functionality, operations, reliability, security, and compliance was split. A development team created an application, an infrastructure team operated it, and a maintenance team took care of reliability. With a DevSecOps approach, on the other hand, a single team is given as much accountability and ownership as possible for all aspects of a product.

In this approach, central enablement teams build shared consumable services for development, infrastructure, and security. They equip these services with guardrails and transparency so that subsequent product developers can use them safely, securely, and efficiently in their operations. Central checks and validations are still performed to ensure that correct procedures and best practices are implemented, but the goal is to make teams accountable for the products they own. This involves providing them with the tools they need to meet their responsibilities—tools that are convenient to use and designed to ensure that the easiest path for developers is also the most secure and reliable route.

Building these consumable services takes time, so companies need to prioritize those with the greatest impact and assign agile teams to build and own them. The software-and-services company decided to prioritize two key services: life-cycle management for its on-premise hosting environments (including provisioning, patching, and so on) and continuous integration and continuous delivery (CICD) pipelines for getting code safely and securely to production. Thanks to these efforts, it managed to cut setup times for hosting environments from three months to 15 minutes, as well as automating deployments that previously required eight hours of manual release work.

Similarly, the financial-services firm focused on creating a secure CICD pipeline, which had to be able to handle the high level of controls needed to satisfy regulatory requirements. Implementing the pipeline cut software release times by half. By introducing mechanisms such as automated security test cases, the firm also streamlined controls by 50 to 80 percent without increasing risk.

**Development and release: Automated pipelines with built-in security controls**

The traditional path to production for software code was lengthy and prone to error. Developers wrote code, and then quality-assurance engineers tested it—usually manually and in settings that bore little resemblance to the environment in which it would eventually be used. Shepherding code through these processes involved many manual steps. Most testing took the form of basic functional and regression tests. Some defects and vulnerabilities slipped through and were caught only after the software was released.

Over the past decade, the DevOps movement has striven to make the path to production more efficient and more effective at catching defects as early as possible, when they are cheaper to address. Leading companies have adopted CICD pipelines to automate workflows and enable best engineering practices to be followed in the writing, reviewing, testing, and deployment of code. In addition to this, DevSecOps companies need to go further by integrating an extra layer of testing into their CICD pipelines to analyze code for potential security issues, run security test cases, and conduct light penetration tests.

To improve their development and release processes, the software-and-services company and the financial-services firm adopted similar approaches. In designing and implementing automated CICD pipelines for deployments to both the cloud and on-premise environments, they took care to involve their security and compliance specialists to ensure that controls would not be compromised in any way. They also used OKRs and metrics dashboards to encourage adoption of the pipelines and increase the levels of automated test coverage.

**Product architecture: Secure by design**

Traditionally, applications have been built from scratch, with all capabilities locked inside a single code package and security not usually tackled until late in the development process. Adding a new feature required extensive testing across the whole system, making upgrades slow and complex to execute. Since applications were seldom designed to scale in a linear fashion as additional load was put on the system, maintaining reliability was often a challenge.

With DevSecOps, by contrast, digital products are conceived and built from the ground up to be secure by design. Security requirements and best practices are factored into all elements of a product, from the code itself to the infrastructure it runs on. Engineers take advantage of existing components built by enablement or shared-service teams, such as container templates and standardized monitoring APIs. They also draw on open-source libraries released by web-scale industry leaders—such as Netflix's Hystrix library for improving fault tolerance—to incorporate best-practice resiliency patterns. Following a "systems of systems" approach, they integrate pre-engineered microservices via loosely coupled interfaces based on well-maintained APIs. All of this improves agility as well as security. Instead of struggling to build their own code, waiting for reviews by security and compliance teams, and iterating until they pass audit checks, product teams reuse code built with expert input and oversight.

Both the software-and-services company and the financial-services firm had legacy systems they needed to support, and both pursued an evolutionary approach to modernizing their application landscape. They used their new processes to build new "greenfield" digital products and incrementally adapted older products to support DevSecOps best practices, including the use of microservices, unit tests, security test cases, static code analyses, and resiliency patterns.

## Common pitfalls to avoid

The best path for an organization to take in adopting DevSecOps depends on many factors, ranging from its size to its familiarity with agile and DevOps methods. But regardless of starting point, all organizations should take care, as they set out on their transformation journey, to avoid a few common pitfalls.

**Pitfall 1: Focusing on tooling alone**

Companies should beware of assuming they can realize the potential of DevSecOps merely by implementing tools such as a CICD orchestration system or a static code analysis tool. To capture the full benefit, they need to make the four shifts described earlier. Without that broader transformation, new tooling is unlikely to be used effectively or consistently across the organization.

**Pitfall 2: Failing to secure leadership buy-in**

If teams are to change their way of working, the leaders of the technology organization must play an active part in steering the transformation. In turn, development leaders must model and reinforce target behaviors and equip their teams to deliver on their new objectives. Finally, security and compliance leaders need to be fully involved to ensure that greater agility doesn't come at the cost of higher risk. To help gain leadership buy-in, successful companies develop a baseline understanding of their agile and DevSecOps maturity and communicate the business case for improvement. Adding key stakeholders to the

team leading the transformation is another way to make leaders feel accountable for ensuring its success.

### Pitfall 3: Focusing only on greenfield development

Though DevSecOps principles are easiest to adopt in new development efforts, applying them more broadly can deliver significant value. For instance, secure CICD pipelines can support larger, more monolithic applications, which can in turn be gradually broken down into loosely coupled microservices. Organizations should constantly explore where they can best deploy DevSecOps to increase agility, security, and reliability.

### Pitfall 4: Taking too long to deliver value

In successful DevSecOps transformations, value is captured from an early stage. Leaders quickly identify any changes that need to be made to product teams and decide which consumable services they need to launch. When rolling out changes, they prioritize products and services that will drive the highest impact or reduce the most risk. By identifying quick wins in the first two or three months, they showcase the value of the transformation and gain support from engineers and the broader organization.

### Pitfall 5: Overlooking capability building and culture

Engineers operating in traditional technology organizations are likely to find adopting DevSecOps a challenge. As well as developing new capabilities, they need to make a major cultural shift by learning to take ownership for their product's security, reliability, and compliance, no matter which team they are on. At the same time, they need to acquire knowledge and skills in creating and operating resilient products to meet their new objectives. Culture and capability building can reinforce one another: when engineers know that the teams they work with expect them to have particular skills, they will be more motivated to develop them.

In addition to avoiding these pitfalls, best-practice organizations ensure they pursue a structured approach to change management. They use dedicated change-management and capability-building workstreams as part of the transformation and roll teams out in waves that allow enough time for training, team alignment, and planning activities, such as agile sprint 0s.[3]

---

Once a niche practice confined to Silicon Valley start-ups, DevSecOps has matured to become a priority for traditional enterprises, too. Adopting the principles outlined in this article will help companies not only acquire the agility to stay current but also strengthen their security to withstand exposure to cyberattacks in the future.

**Santiago Comella-Dorda** is a partner in McKinsey's Boston office, **James Kaplan** is a partner in the New York office, **Ling Lau** is a partner in the New Jersey office, and **Nick McNamara** is an associate partner in the Chicago office.

The authors wish to thank Nagendra Bommadevara, Daniel Brosseau, Alharith Hussin, Steve Jansen, Jesus Mathus Garza, Mark Mintz, Thomas Newton, Jan Shelly Brown, Marc Sorel, Kevin Telford, and Charles Wisniewski for their contributions to this article.



[3] Agile sprint 0s are short efforts before a team starts working together, used to create a common vision, align on team norms, and create a product backlog for what they plan to build.