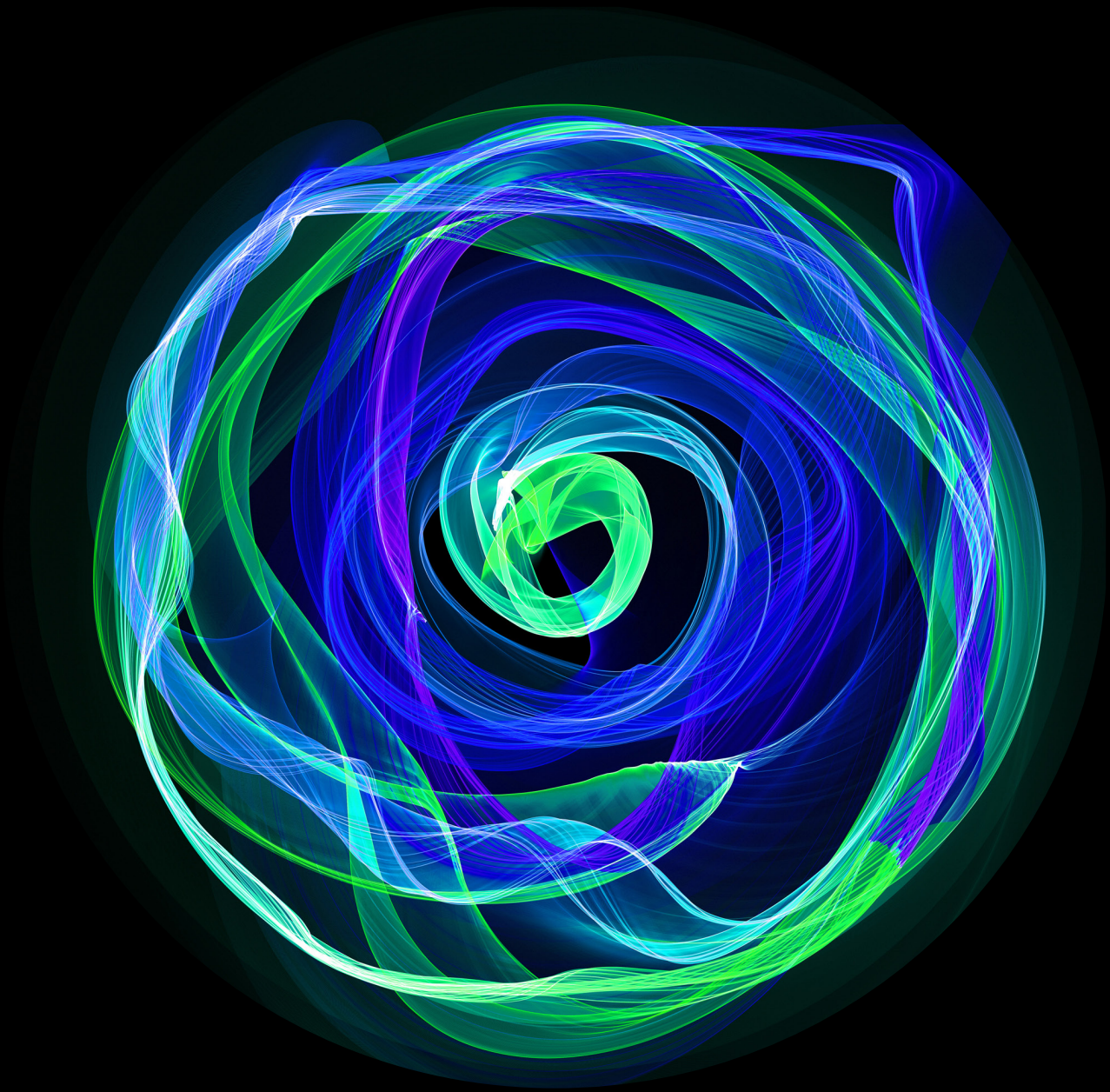
Deloitte.



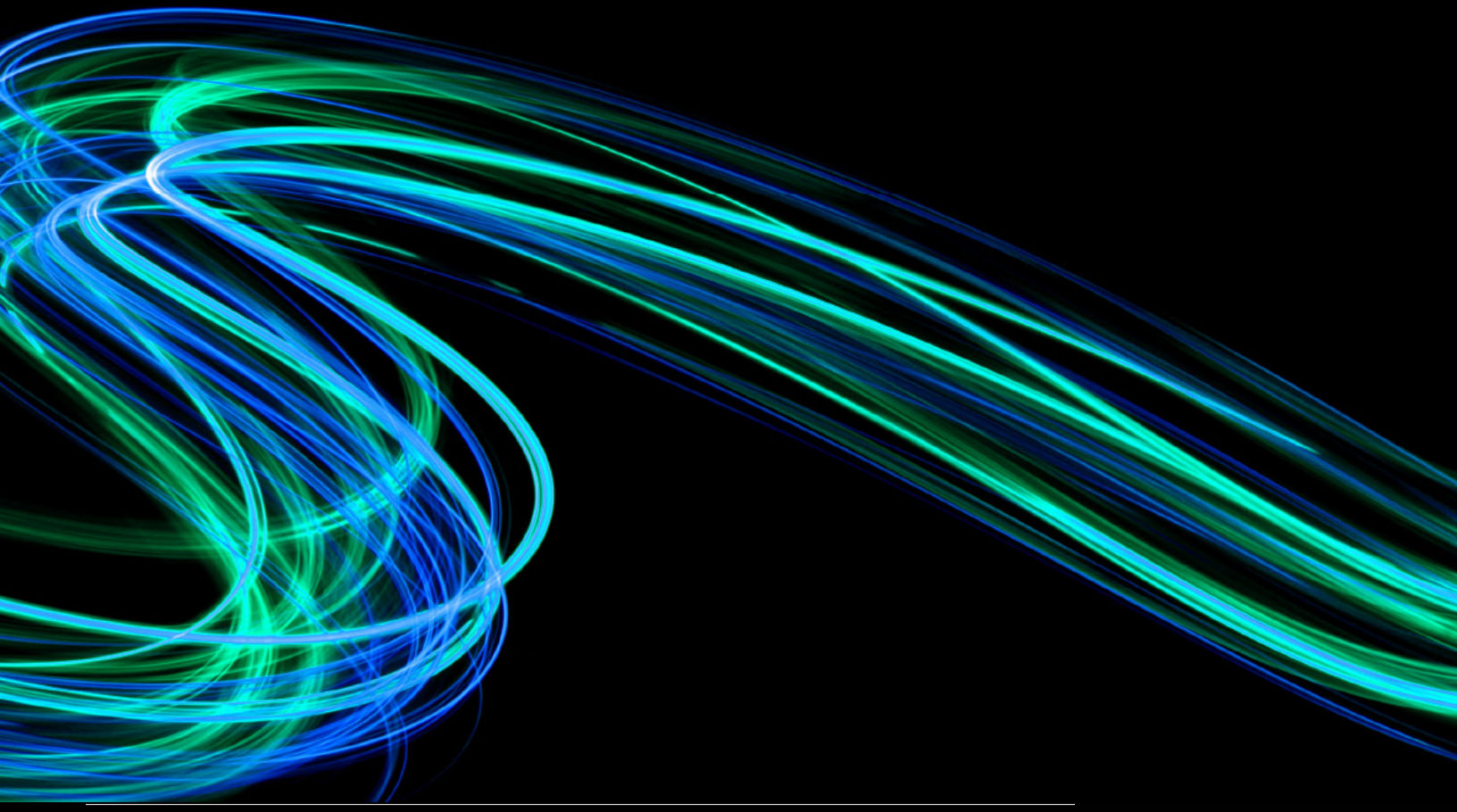
Auditing DevSecOps Projects

"Continuous everything" calls for a new approach to mitigating information technology (IT) risks

In the ever-evolving world of IT development, there is once again a new kid on the block.

Many companies are on the journey to employ Development Security Operations (DevSecOps)[1] as an evolutionary extension of Agile principles (refer to our point of view on Auditing Agile Projects for more information). As noted in a recent Gartner publication ("Hype Cycle for Agile and DevOps," 2020), "DevOps continues to grow, and the percentage of respondents saying they do not have any plans to adopt DevOps has dropped from 28% in 2016 to only 8% in 2019." Wow—this is a game changer!

"New technologies and approaches are being introduced on an increasingly frequent basis."[2] Like the shift from traditional waterfall development to Agile, the growing movement toward DevSecOps has significant implications for internal audit (IA) teams. Change management processes are continuous and largely automated in a DevSecOps environment, which challenges IA teams to shift their mindsets about IT risks and the controls in place to mitigate them. The first step in adapting to a DevSecOps world is to better understand what DevSecOps is and what it isn't.



1. Although DevOps and DevSecOps should not be used interchangeably, the control considerations within this POV apply to both.
2. George Spafford and Joachim Herschmann, "Hype Cycle for Agile and DevOps," Gartner, 2020.

DevSecOps defined

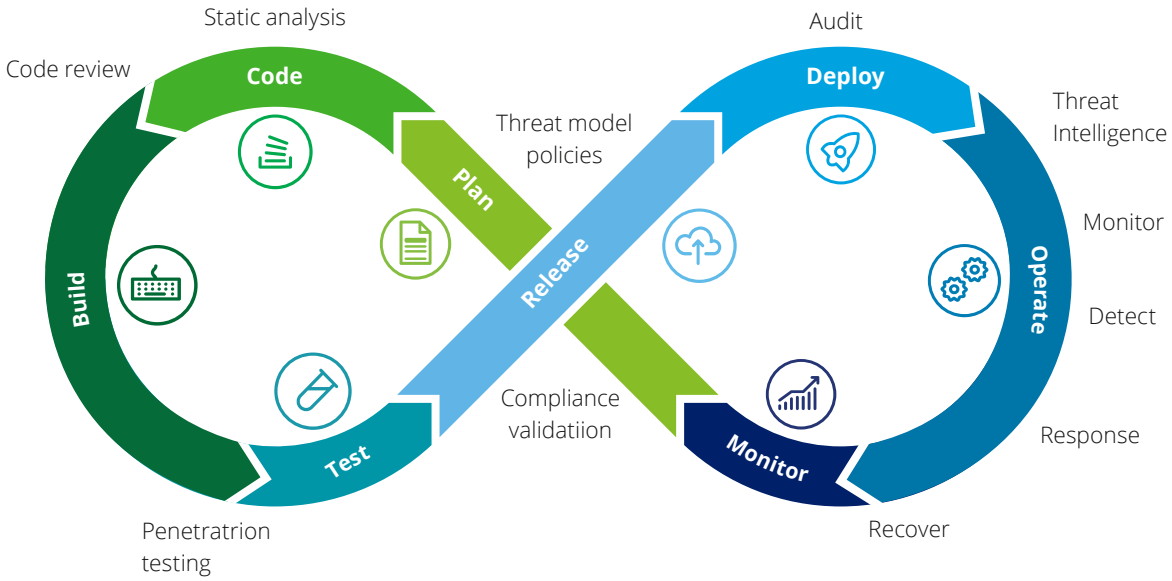
At a high level, DevSecOps is a software development and delivery approach that emphasizes communication and collaboration between development, security, and IT operations (ITOps). More than a methodology, DevSecOps is also a mindset that builds on Agile and Lean thinking to provide technology faster, with greater stability, quality, scalability, and security.

Even though DevSecOps is a combination of the words "development," "security," and "operations," the term encompasses many teams involved in the software development and delivery process. "Dev" is understood to mean everyone on the code development side, including developers, front-end designers, and quality assurance. Meanwhile, "Ops" is understood to include everyone on the systems side, including system administrators and support teams responsible for the product after it's been moved to production. "Sec" is understood to mean cybersecurity professionals responsible for system restriction, compliance, and secure applications.

In a DevSecOps world, the collective team works together to support development, delivery, and post-go-live maintenance, using automation and monitoring to build, test, and release software rapidly, frequently, and more reliably.

Historically, developers have seen security professionals as causing delays, and security professionals have seen developers as responsible for introducing security flaws subject to compromise. These teams coming together drives increased value and efficiencies by solutioning together up front.

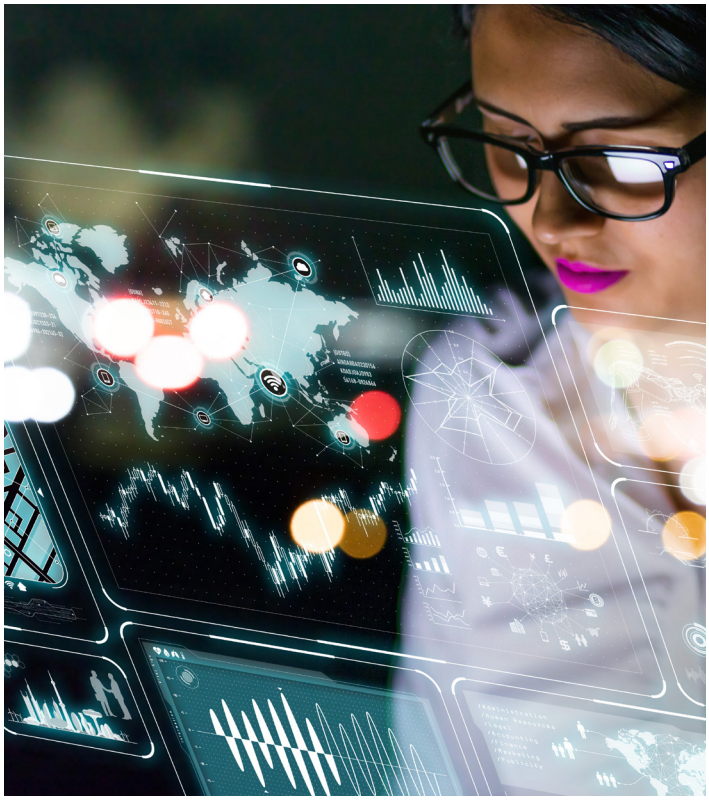
DevSecOps: Integrate security into DevOps[3]



3. Abhishek Baranwal, "DevSecOps: Security with DevOps," https://blog.knoldus.com/devsecops-security-with-devops/, last modified September 19, 2019

Risk management changes

Like Agile, DevSecOps offers the advantage of short, frequent releases. Indeed, the primary goal of continuous delivery with DevSecOps is to make software deployments painless, low-risk events that can be performed incrementally, at any time, and on demand. However, in a DevSecOps world, team members wear many hats, and tools are used to automate historically manual tasks, such as code quality checks, execution of test scripts, and deployments. These factors raise some questions about the efficacy of traditional change management controls in a DevSecOps environment.



Characteristics of DevOps

DevOps dissolves the barriers between development and operations to generate value quickly with quality and stability. It can:

- **Deliver** software faster with less effort by optimizing the end-to-end technology value stream
- **Continuously** provide high quality, from the point of creation to operation
- **Reduce** manual rework and heighten quality through automated delivery of software and infrastructure.
- **Make** work in progress more visible to better understand constraints and balance workloads.

Okay, auditors, let's rethink historical segregation of duties (SoD)!

DevSecOps teams are different than what many organizations are accustomed to, with several team members wearing multiple hats as compared with the traditional development life cycle. This challenges IT teams to reconsider the effectiveness of traditional change management controls, such as SoD.

The risk of a user making an inappropriate change to the environment, either maliciously or unintentionally, without appropriate testing and approval has historically been mitigated by SoD between the person who develops the change and the person who implements it, as well as appropriate testing prior to deployment. With more frequent, iterative changes and the multi-skilled roles within the DevSecOps team, traditional SoD isn't maintained in a DevSecOps environment.

Increased use of tools in a DevSecOps environment may also inadvertently create SoD conflicts if not carefully understood and designed. Consider a company that has integrated DevSecOps into its change management process and that manages its DevSecOps tools through a centralized team. In this instance, the same administrators may inadvertently have the ability to both develop and promote code across tools.

When SoD isn't possible, DevSecOps teams should consider identifying other controls to address the risk that a single user could make a change without appropriate testing and approval. Some common examples include:

Requiring changes to be approved by someone else on the team, ideally via automated workflows. Some examples of this may be user acceptance testing (UAT) performed by the business, product owner approval, or business stakeholder approval. Regardless of how it's implemented, the idea is to prevent one person from being able to change the code and put it into production without anyone else knowing or being involved.

Mitigating controls, such as monitoring and alerting, to detect inappropriate changes that did not follow standard procedures. This can be done manually or, for more mature organizations, via bespoke continuous monitoring rules or alerts that show when a change was made and how it ties back to the other controls in the process.

Handling the tools

Automation can offer many benefits, such as continuous monitoring and workflow management. Teams often rely on tools to achieve these goals. For instance, DevSecOps teams may likely use tools to scan source code and to automate regression, security, and performance testing. Such tools can enable developers to identify and address issues like security vulnerabilities that previously wouldn't have been detected until after the product had been moved into production.

By building an automated pipeline, these activities can be performed continuously throughout the delivery process, so quality is built into products from the beginning.

However, this new way of working also poses new risks. A lot more tools come into play in an automation-heavy DevSecOps environment, which means there are risks around those tools being modified inappropriately or configured incorrectly. Automated controls also include configurable settings, as well as automated rules and/or algorithms and calculations, which introduce additional change management challenges.

For instance, an unauthorized user could modify the information in a tool that is used to assess the effectiveness of a control or that could be used as evidence in an audit. Or, since tools can include automated controls that rely on general IT controls (GITCs), an unauthorized change could cause the tool to malfunction and result in an ineffective control.

Managing the manual aspects

Automation doesn't mean that humans are left out of the process. Manual decisions still need to be made to tell the automated tool how to perform. These human-centric aspects of the process should also be considered in the risk-management approach. For example, in order to gain assurance that the code is functioning as intended, someone will need to know which test scripts are relevant and how to incorporate appropriate scenarios and acceptance criteria into them. Similarly, if automated workflows are being used, someone will need to determine to whom those tasks will be routed and how the flow will be maintained. And, if continuous monitoring is being implemented, someone will need to consider what or who should be monitored and whether the process is accurate and complete.

Quick tip!

Automated controls largely depend upon effective GITCs to govern consistent operations, and the pipeline itself depends upon automated controls to function efficiently and consistently. Accordingly, risks can compound quickly when handling automated tools, such as those used for automated testing. For instance, an automated testing tool typically needs to be configured so it can recognize when acceptance criteria have been met. A simple configuration mistake could not only prevent the test script from running, but also cause one or more GITCs to fail. Furthermore, if test scripts are inappropriately modified, or the team fails to update them when new functionality is introduced, it could cause similar cascading effects.

Conclusion

Bringing development and operations together into a cohesive approach opens the door for software development teams to attain a new level of efficiency and effectiveness. When auditing, the intent is to help them to walk through that door while addressing the associated change management risks. This requires knowledge of how DevSecOps works, as well as a shift in perspective. In a DevSecOps environment, historical project and change management controls should be reevaluated because the ways of forming teams, working together, and deploying products are different.

By focusing on the new change management risks and how processes can be tweaked, DevSecOps teams can leverage these tools to mitigate risks without bogging down development by introducing cumbersome new compliance processes.

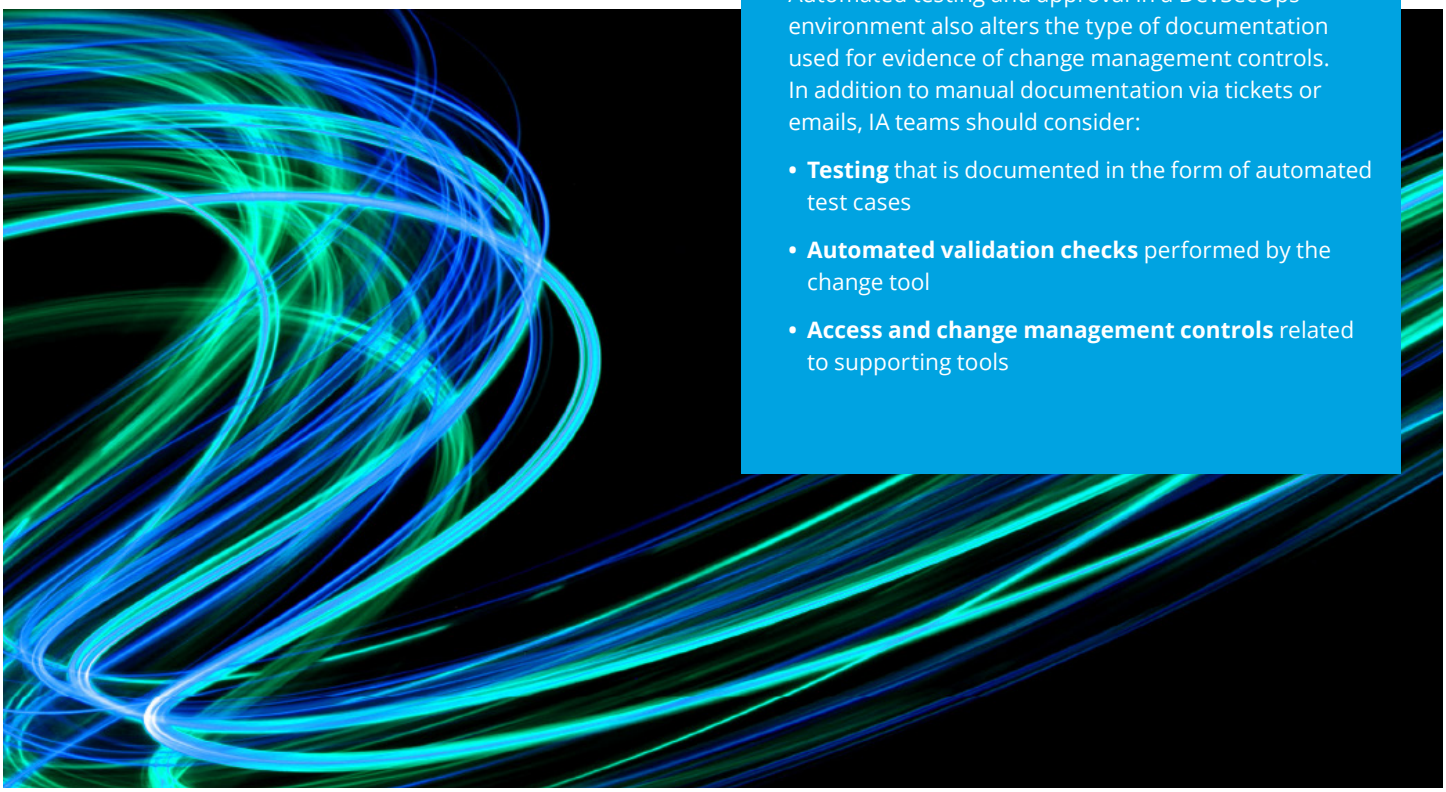
If IA can bring this point of view and mindset to DevSecOps-related audits and consultations with technical teams, compliance and DevSecOps efficiencies can harness value creation.

Putting DevSecOps to the test

In a DevSecOps environment, IA teams will increasingly need to bring tools into the audit scope, testing automations that function as controls, as well as testing the controls over source data. At a minimum, IA teams should consider the tools that support GITCs in managing approvals, generating reports, and maintaining documentation. This includes understanding the tool, assessing how it is used, considering the risks, and determining any additional testing procedures that may be required. Pay attention to tools used within the change management process. For example, if a code management tool is used to control access, the GITCs over it may likely be relevant to an audit.

Automated testing and approval in a DevSecOps environment also alters the type of documentation used for evidence of change management controls. In addition to manual documentation via tickets or emails, IA teams should consider:

- **Testing** that is documented in the form of automated test cases
- **Automated validation checks** performed by the change tool
- **Access and change management controls** related to supporting tools



Contact us

Sarah Fedele
Principal
Deloitte & Touche LLP
+1 713 982 3210
sarahfedele@deloitte.com

Ranjani Narayanan
Senior manager
Deloitte & Touche LLP
+1 617 437 3847
rnarayanan@deloitte.com

Ryan Gentry
Manager
Deloitte & Touche LLP
+1 713 982 2298
rygentry@deloitte.com

Sarah Adams
Managing Director
Deloitte & Touche LLP
+1 713 982 3416
saradams@deloitte.com

Kristen Heikkinen
Senior manager
Deloitte & Touche LLP
+1 617 437 3488
kheikkinen@deloitte.com

Julia McDonald
Manager
Deloitte & Touche LLP
+1 617 585 5961
jumcdonald@deloitte.com

Deloitte.