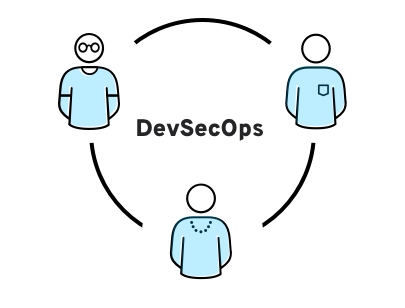
DevOps isn’t just about development and operations teams. If you want to take full advantage of the agility and responsiveness of a DevOps approach, [IT security](https://www.redhat.com/en/topics/security) must also play an integrated role in the full life cycle of your apps.

Why? In the past, the role of security was isolated to a specific team in the final stage of development. That wasn’t as problematic when development cycles lasted months or even years, but those days are over. Effective DevOps ensures rapid and frequent development cycles (sometimes weeks or days), but outdated security practices can undo even the most efficient DevOps initiatives.

Now, in the collaborative framework of DevOps, security is a shared responsibility [integrated](https://www.redhat.com/en/solutions/agile-integration) from end to end. It’s a mindset that is so important, it led some to coin the term "DevSecOps" to emphasize the need to build a security foundation into DevOps initiatives.



DevSecOps means thinking about application and infrastructure security from the start. It also means [automating](https://www.redhat.com/en/topics/automation) some security gates to keep the DevOps workflow from slowing down. Selecting the right tools to continuously integrate security, like agreeing on an [integrated development environment (IDE)](https://www.redhat.com/en/topics/middleware/what-is-ide) with security features, can help meet these goals. However, effective DevOps security requires more than new tools—it builds on the cultural changes of DevOps to integrate the work of security teams sooner rather than later.

Whether you call it "DevOps" or "DevSecOps," it has always been ideal to include security as an integral part of the entire app life cycle. DevSecOps is about built-in security, not security that functions as a perimeter around apps and data. If security remains at the end of the development pipeline, organizations adopting DevOps can find themselves back to the long development cycles they were trying to avoid in the first place.

In part, DevSecOps highlights the need to invite security teams at the outset of DevOps initiatives to build in information security and set a plan for [security automation](https://www.redhat.com/en/topics/automation/what-is-security-automation). It also underscores the need to help developers code with security in mind, a process that involves security teams sharing visibility, feedback, and insights on known threats. It’s possible this can include new security training for developers too, since it hasn’t always been a focus in more traditional application development.

What does built-in security really look like? For starters, a good DevSecOps strategy is to determine risk tolerance and conduct a risk/benefit analysis. What amount of security controls are necessary within a given app? How important is speed to market for different apps? Automating repeated tasks is key to DevSecOps, since running manual security checks in the pipeline can be time intensive.

## **DevOps security is automated**

To do: Maintain short and frequent development cycles, integrate security measures with minimal disruption to operations, keep up with innovative technologies like [containers](https://www.redhat.com/en/topics/containers) and [microservices](https://www.redhat.com/en/topics/microservices), and all the while foster closer collaboration between commonly isolated teams—this is a tall order for any organization. All of these initiatives begin at the human level—with the ins and outs of collaboration at your organization—but the facilitator of those human changes in a DevSecOps framework is [automation](https://www.redhat.com/en/topics/automation/whats-it-automation).

But what to automate, and how? There is [written guidance](https://itrevolution.com/book/devops-and-audit/) to help answer this question. Organizations should step back and consider the entire development and operations environment. This includes source control repositories, container registries, the [continuous integration and continuous deployment (CI/CD) pipeline](https://www.redhat.com/en/topics/devops/what-cicd-pipeline), application programming interface (API) management, orchestration and release automation, and operational management and monitoring.

New automation technologies have helped organizations adopt more agile development practices, and they have also played a part in advancing new security measures. But automation isn’t the only thing about the IT landscape that has changed in recent years—[cloud-native](https://www.redhat.com/en/topics/cloud-native-apps) technologies like containers and microservices are now a major part of most DevOps initiatives, and DevOps security must adapt to to meet them.

## **DevOps security is built for containers and microservices**

The greater scale and more dynamic infrastructure enabled by containers have changed the way many organizations do business. Because of this, DevOps security practices must adapt to the new landscape and align with [container-specific security guidelines](https://csrc.nist.gov/publications/detail/nistir/8176/final).

Cloud-native technologies don’t lend themselves to static security policies and checklists. Rather, security must be continuous and integrated at every stage of the app and infrastructure life cycle.

DevSecOps means building security into app development from end to end. This integration into the pipeline requires a new organizational mindset as much as it does new tools. With that in mind, DevOps teams should automate security to protect the overall environment and data, as well as the continuous integration/continuous delivery process—a goal that will likely include the security of microservices in containers.

Get expert perspectives on security throughout the container application stack and life cycle with this webinar series.