6 common pitfalls for DevOps teams and how to avoid them

1. Lack of alignment.

A strong DevOps team is only successful if it's truly crossfunctional. It may be tempting to start small, within your own engineering team, but then you run the risk of embracing principles within a silo—whichis the antithesis of what a DevOps strategy aimsto do. As the saying goes: Teamwork makes the dream work.

How to avoid: Begin by doing the foundational work to ensure that all key teams are on board. This begins at the top, so that leadership can help steer and champion teams to work together and improve processes across the entire software development lifecycle. Stakeholders may include orgs such as IT, UX, product, and security.

2. Testing shortcuts.

Automation may seem synonymous with DevOps, but adopting and implementing a DevOps strategy and team extends well beyond automation of workflows. That said, creating an environment of continuous testing, to enable continuous delivery, is a core tenant of a DevOps practice. It's important not to skimp on resources or skip setting up the test automations that seem too time-consumingor complex.

How to avoid: Invest in building out your test automation suite so that it runs with each commit. Your DevOps solution should help to enable this process, ideally with the help of AI to identify potential issues and suggest solutions, so that you don't add to your backlog of security debt. Enterprise-grade solutions like GitHub Copilot can help you code faster and remediate in real time. Copilot can also help with generating unit and integration tests.

3. Complicated tool integrations.

More tools add up to more complexity. Your DevOps toolchain should include applications for things like source control, CI/CD, testing, infrastructure provisioning, and even notifications—and they should talk to each other. But today, this shouldn't be a manual process or require custom scripts to tie the pieces together—especially because there are comprehensive, unified platforms available to help reduce and simplify your toolchain. The more that you can do within the developer workspace, the better.

How to avoid: Begin by performing a tool audit. The goal isn't to make what you already have work at any cost (and then add more). Consider what you really need and which solutions are available now to help make your teams' lives easier. GitHub Actions, for example, automates your software workflows,with CI/CD, so you can build, test and deploy your code right from GitHub.

4. Workloads and burnout.

Development teams often feel overwhelmed and overworked, often because they're asked to do increasingly more outside of their job description. A scarcity of security professionals relative to the number of developers puts pressure on developers to become experts in application security. This is not a small ask, and it's an issue felt by developers around the globe. According to the World Economic Forum, there is a global talent shortage of cybersecurity professionals. The more that organizations can adopt software and DevOps practices to help bridge the gap between development and application security, the better. The orgs who do will have an advantage.

How to avoid: Look for DevOps tools and solutions that can help offload tasks for developers (and security teams), helping to reduce burnout, turnover, and friction between teams. Developers typically face coding or remediation learning curves, and spend too much time on rote coding tasks and application security testing requirements, leading 81% of developers to release vulnerable code under pressure to ship. Bringing in an AI pair programmer can help remove boilerplate tasks so that developers can focus on the coding that matters most, and is the most fulfilling.

CONTINUED NEXT PAGE →

5. Stuck on points of failure.

Your DevOps practice should create a more failure-tolerant environment, but that doesn't mean it will be failure-free. Testing and learning is part of the game, but the goal is to adopt a DevOps solution that allows you to test and learn faster. Don't make the mistake of letting a failure slow you down or point fingers at a particular team. DevOps is, if anything, a #oneteam goal.

How to avoid: Failures create learning opportunities. If your process isn't going according to plan, take a second look at the workflow and allow space for teams to identify points of failure and opportunities to innovate and improve. Test new ideas and continueto learn.

6. A point solution approach.

The flexibility of DevOps can be both a blessing and a curse. By design, it gives individuals in an organization more power and autonomy. In more chaotic environments, however, poorly vetted features across multiple products can lead to code that's deployed, amended, or even rolled back, causing customer and developer frustration.

How to avoid: Just as you need both leadership and crossfunctional alignment to support your DevOps strategy, you also need alignment around your DevOps platform and its implementation. Be sure to designate approval points and controls across processes and leave room for some flexibility.

According to Forrester,

"It's obvious that solutions that merge continuous integration, continuous delivery, and release automation are valuable. Enterprises are eager to adopt them — they'd rather have a cohesive tool that they can run out of the box than assemble things themselves with scripts-and-glue code."

DevOps can streamline software development and delivery, and GitHub offers a unified platform for your DevOps lifecycle. As the world's most widely adopted AI-powered developer platform, used by 100+ million developers around the world, you'll be in good hands and good company.

Get started today. Learn more at https://github.com/enterprise.