6 DevOps pitfalls and how to avoid them

DevOps is a transformative practice. Companies in almost every industry are increasingly adopting DevOps to give teams the time and freedom they need to tackle more challenging projects. But whenever you embrace a new strategy, there can be hiccups. Here are some common issues developers and organizations run into when adopting DevOps—and how to overcome them.

1. Not getting all your teams on board

It's possible to get DevOps up and running within your software organization and yet still not see the speed of software delivery improve. Why? It may be because you only implemented DevOps principles in the engineering department—or worse yet, created separate DevOps teams—while keeping everything else in your company's organization chart the same.

Bottom line: DevOps requires wholesale organization change and everyone's buy in. Teams outside of engineering need to adopt DevOps too, including IT, UX, product, and marketing.

2. Limited testing

Setting up test automation can be an intense, timeconsuming process. This can mean some teams run certain more complex tests manually. That's a mistake. If you don't invest in building out your test automation suite, you won't be able to run your entire test suite with each commit. This can leave bugs and issues undiscovered until later in the workflow, making them harder to fix.

Bottom line: Continuous delivery requires continuous testing. Dedicate time and resources to build a strategic automated testing suite to set yourself up for long-term DevOps success.

3. Incomplete tool integration

More tools mean more complexity. Your DevOps toolkit contains applications for things like source control, Cl, deployment, testing, infrastructure provisioning, and even notifications. What are the odds they all talk to each other? Many software organizations end up managing their DevOps toolchains manually or using custom scripts to tie everything together—an approach that becomes less sustainable as more tools and use cases are added.

Bottom line: Take advantage of GitHub Actions, a Cl/ CD and automation tool that allows workflow steps to be treated as code, tying in integrations as needed.

4. Workload overload

One good reason many companies move to a DevOps model is their development teams are overworked. But an excessive workload for too few people can also cause a DevOps implementation to fail. Introducing new tools and processes to a team that's already struggling is a recipe for chaos, employee burnout, and higher turnover.

Bottom line: Re-prioritize workloads and focus on core goals and objectives where possible before attempting the DevOps transition.

5. Unwillingness to fail

DevOps creates a more failure-tolerant environment, but that doesn't mean it's failure-free. In the aftermath of a failure, many beginner DevOps organizations make the post-mortem mistake of assigning blame to a single point in their workflow.

Bottom line: Treat failure as a learning opportunity and create space for people and teams to identify failure points across your workflows.

6. Total product anarchy

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 and redesigns can be deployed, amended, or even rolled back, causing customer frustration.

Bottom line: Before implementing DevOps, design approval points and controls in your processes, and make sure you have up-to-date documentation.

DevOps can streamline software development and delivery, bringing untold benefits to your business. Trust GitHub, the software development platform used by 83 million developers worldwide, to help you implement the DevOps model while avoiding these pitfalls.