# SIX DEVOPS PITFALLS FOR ENGINEERING LEADERS TO AVOID

DevOps can be a transformative practice for businesses of all sizes and types. Companies in almost every industry are using DevOps to give teams the time and freedom to tackle more challenging projects. As with all software development strategies, there are some shared pitfalls to watch out for.

## 1. Partial org transformation

It's possible to get DevOps up and running within your software organization and yet still not see any overall improvement to the speed of delivery. In many cases, this is due to implementing DevOps principles only in the engineering department while keeping everything else in your company's org chart the same. Since DevOps is a more agile software delivery model, teams outside of software need to align to match, including IT, UX, product, and marketing.

## 2. Incomplete testing automation

Continuous delivery also means continuous testing. Setting up test automation can be an intense, time-consuming process, which means some teams leave certain complex tests alone to be run manually. If you go this route, you won't be able to run the entire test suite with each commit. At best, you can run only a core set of tests upon commit, leaving the full test suite to be run periodically. This can leave bugs undiscovered until later in the workflow, making them harder to fix.

## 3. Tool integration problems

Your DevOps toolkit contains applications for things like source control, CI, 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 toolchain manually or using custom scripts to tie everything together—an approach that becomes less sustainable as more tools and cases are added. To solve this problem, GitHub recently released a new feature called **GitHub Actions** that allows workflow steps to be treated as code, tying in integrations as needed.

## 4. Too much too fast

One good reason many companies move to a DevOps model is that their development teams are overworked. But an excessive workload can also cause a DevOps implementation to fail. Introducing new tools and processes to a team that's already struggling to manage the workload is a recipe for chaos, employee burnout, and higher turnover. Re-prioritize, defer, or delegate work to contractors where possible before attempting the DevOps transition.

## 5. Unwillingness to fail

DevOps creates a more failure-tolerant environment, but that doesn't mean failure-free. In the aftermath of a failure, many beginner DevOps organizations make the post-mortem mistake of assigning blame to a point in the workflow. Instead, treat failure as a learning opportunity. Taking an issue-solving approach is much more useful without introducing extra stress. Some companies, Netflix among them, actually cause simulated failures on purpose in order to get teams used to handling them.

## 6. Total product anarchy

The flexibility of DevOps can be both a blessing and a curse. By design, it gives DevOps teams more power and autonomy, but those teams might also end up doing things they shouldn't. In more chaotic environments, poorly-vetted features and redesigns can be deployed, amended, or even rolled back, causing customer frustration. Before implementing DevOps, it pays to carefully design some approvals and controls into your processes.

DevOps has the power to streamline software development and delivery, bringing untold benefits to your business. GitHub, the software development platform used by 30 million developers worldwide, can help you implement the DevOps model while avoiding these pitfalls.

**Get started with GitHub today.**

**Visit https://github.com/enterprise**