split®

De-Stressing Feature Releases for Developers

Contents

Introduction

As a developer, you've experienced the excitement, joy, and accomplishment of bringing an idea to life and having it function as it should. It's part of what drives you to constantly learn and experiment. It's what inspires you to push the limits of what's possible. It's why you do what you do.

But the perks of being an innovator in product development come with pitfalls. There's fatigue from long hours dedicated to problem-solving and disappointment when missing out on social and family events due to work. As with most things, product development is more journey than destination. So, what makes it all worthwhile?

Developers like you deserve an experience that gives you autonomy and minimizes stress.

- You should be enabled to deliver high-quality code and still have the time and energy to do things outside of work.
- You should collaborate with, learn from, and earn the respect of your equally talented peers.
- You should feel valued by your organization and given the space to tackle problems with creative ideas.

In today's tech environment, advocating for these key elements of the developer experience isn't easy. With more companies downsizing their teams and halting their hiring efforts, you may feel like you don't want to rock the boat—even if you're sinking. You don't want to look ungrateful by requesting more autonomy, better processes, and additional resources.

The truth is, downturn or not, the development work that so many organizations rely on takes creativity. And as a developer, it's unlikely you'll produce your most inspired solutions if you aren't supported to do so.

To work autonomously, ensure quality, find balance, and collaborate effectively, you need tools and processes that will set you up to excel. In this eBook, we're taking a closer look at four points of friction causing developers stress and the tooling that can take the pressure off the feature release cycle.

The Past and Future of Feature Release

First, let's take a look back. Software development teams once operated under the waterfall model, with hundreds of developers writing application code that would eventually be packaged together. Within this approach, team members had discrete roles. Some were responsible for developing code, and others were responsible for pushing it to production. Pieces of waterfall's legacy, such as dedicated QA teams and late-stage security evaluations, remain in the industry today, but we have largely moved on.

This way of doing things created an excessively high-pressure environment for developers. With work occurring in silos, individual contributors lacked autonomy. Their progress stagnated when they needed to wait on other teams. Then once they handed off their contribution, they had little say in what happened to it next. Release cycles were slow and infrequent. When a release does occur, it was a momentous event with all hands on deck in case something went wrong.

Critics who perceived that the scale and rigidity of waterfall weighed down development cycles sought more productive, lightweight methods that could be applied by leaner teams. Their explorations led to the publication of the Agile Manifesto. This catalyzed the transformation of development best practices to favor iteration, and the advent of DevOps and continuous delivery models. To support agility, practices like feature management were introduced, making it easier for teams to separate the code into manageable parts that could be worked on and deployed independently — and with less risk.

Now, with the right tools and frameworks, feature releases can occur multiple times a day, and many developers feel equipped to work more independently, under less unnecessary pressure. What does this look like in practice? Can developers really move fast and thrive?

Solving for Feature Release Friction

Today's developers face numerous challenges related to feature releases. This includes the mounting focus on business needs that can send developers into burnout territory. It doesn't have to come to that. Here are four pain points that can be tempered with tooling.

Pain Point #1: A Demanding Release Schedule

As a developer, you probably have many horror stories about late-night and weekend deployments. Times when a new feature broke the application and had to be completely rolled back. Or when everything looked like it was going fine... until it wasn't. You may even have a story or two of a time when everything went off without a hitch.

These off-hours deployments are a familiar experience. Ultimately, releasing features during low-traffic periods reduces the potential impact if something goes wrong — be it a security vulnerability, an issue that didn't present itself in the testing environment, or a scalability concern. But they also require developers to be on-call, pulled away from time with family or awake in the middle of the night. It's not sustainable, and it keeps you from being at your best.

Companies are on a mission to kill the notion of release nights or weekends, changing the philosophy of product releases. As a result, releases become more segmented, iterative, and don't require people to give up their time away from work. Why would you ask your developers to make that kind of sacrifice when there are tools available that make it unnecessary?

Along with a mindset shift, feature flags equip engineers to easily test changes with select users. They can then quickly undo any problematic features, thus reducing the potential impact of a change. This way, most changes can happen during working hours, and developers can release as often as they want — even during peak customer hours.

Pain Point #2: Looming Threats of Errors and Downtime

Causing an outage or releasing an error into production is the stuff of nightmares for engineers. This is especially true today, as customers have such high expectations of their experiences with tech and have no problem shaming brands on social media.

Outages can cause loss of revenue, reputational damage, or security implications for users and the company alike. Behind the scenes, a problematic release can lead to further issues down the road if not remedied properly. This is a big (and unfair) burden for you as a developer to carry on your shoulders. It hampers your ability to test new ideas out of fear of breaking something.

Feature management offers an alternative to large deployments that risk major impact. It allows developers to do targeted testing to just a few users at a time. Therefore, they can account for real use cases that might not have made it into the pre-production environment. You can even test with internal users, adding a layer of testing in production before full launch.

The kill switch mechanism is also useful here – turning off a flag reverts the feature immediately, vastly reducing downtime potential if there's an issue. Paired with data monitoring, that means it's possible to identify issues just as quickly as you can disable them.

For developers, feature management instills confidence. You can build on new ideas without the fear of breaking things, and that's priceless.

Pain Point #3: Slow Release Cycles Impact Workload, Consistency, and Motivation

Developers want to be able to move fast, delivering on new product features at scale. As a result, the company can meet the demands of its customers and lead the way in its industry. What holds you back is risk.

Not only that, you want to see your work go live and get feedback on how it performs. The problem is that without agile practices and the tools to enable them, release cycles are complex and slow.

For example, an organization launches a solid product that gains traction in the market. The engineering team grows and the codebase becomes more complex. However, by the time they're a few years and product features in, releases become unwieldy. You make a change, then it goes through a long manual testing process with a separate quality assurance team. Figuring out how a new change impacts the full user experience can take ages since it's not always possible to identify the problem area.

Rapid releases don't have to be wishful thinking, though. Feature management tools allow engineers to deploy small feature changes iteratively. Features can be tested individually, which reduces the need for lengthy reviews. With the addition of access to granular monitoring data, quality assurance can be almost instantaneous.

Pain Point #4: The Ongoing Desire for Autonomy and Freedom

Developers are big thinkers. Given the time and space, you would probably happily contemplate ways to make things better, faster, and more efficient. Organizations would benefit from letting their developers have free reign to ideate and test, if they weren't so worried about breaking something.

Organizations can enable developers to tap into their brain power and execute on their ideas without introducing any risk to the operation. A modern feature flag and monitoring tool speeds up the release cycle and reduces the burden on developers. So you get more time to focus on your areas of specialization. It allows engineers to make constant changes and gives you the freedom to be creative. Not to mention, it also enables you to test ideas in a risk-averse way, opening the door to new innovations for the product.

Dispelling the Myth: Going Faster Isn't Always Riskier

The best developers pride themselves on delivering good code without cutting corners. You have to go slow to go fast. You have to follow processes and documentation and go through robust code reviews. Then, you have to test in various environments to ensure the code is high quality, reducing the risk of any problems.

A feature that's whipped up too quickly could break the rest of the application — and that could impact how the user experiences the product. Then again, speedy progress could also allow you to stay on schedule, go home on time, and harness flashes of inspiration to expand your impact. This is what a unified feature flag strategy can enable.

Feature flags enable trunk-based development, where smaller iterations of code can be written and reviewed independently. This cuts down the review time, making it quicker and easier to ship code.

Beyond shortening the development lifecycle, feature flagging platforms equipped with monitoring, measurement, and segmentation capabilities also mitigate risk. Performance issues can be identified immediately, and features can be instantly rolled back without impacting the rest of the application. There's a lot of value in this — it allows for a culture of innovation and experimentation.

While testing environments are vital elements of the development lifecycle, they can't always account for all the scenarios that require production. Feature flags thus allow for an extra layer of testing to confirm quality. By leveraging segmentation, you can test in production for internal teams or small groups of customers and limit the breadth of potential issues.

If we look ahead at the future of software development, we envision spaces where developers can test, fail, and learn without fear. Developers will be visited often by that good feeling, the one you get when you actively contribute high-quality code and still have time to pursue creative ideas that change the face of their product. That's the future we're aiming for with feature management.