PagerDuty

5 Ways to Empower Developers

Maximize your development team’s impact

In this guide, we'll share some benefits that development teams realize when they are tasked with managing their services in production. We'll also share best practices around how to make operating responsibilities as stress-free as possible, so your teams can spend more time on coding and delivering higher-quality services, instead of on fixing issues.

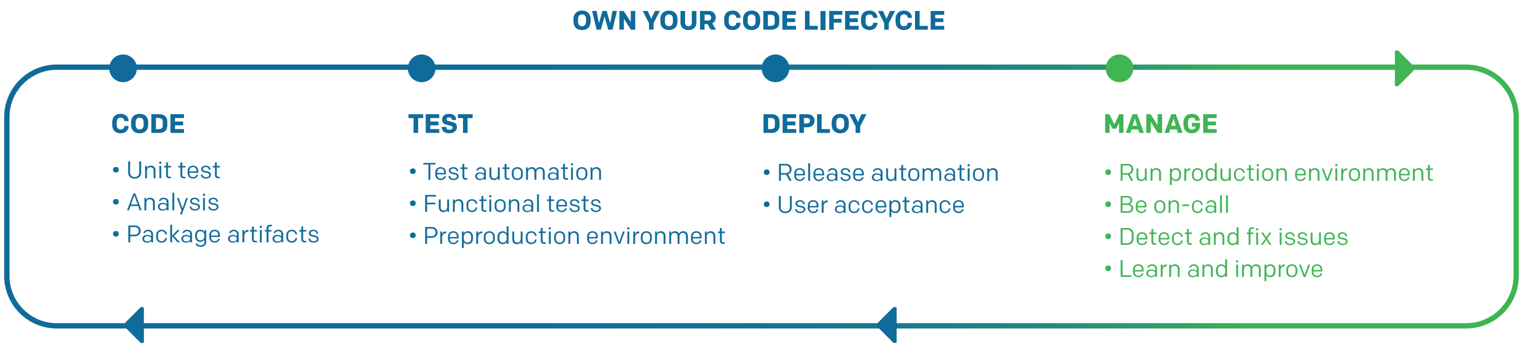
Contents

PagerDuty

The Developer's Role is Changing

Today, more than ever, developers want and need to be accountable for the customer experience. DevOps and the proliferation of new technologies such as microservices, containers, and hybrid cloud infrastructure, are transforming the speed at which development teams can deliver customer value. Teams taking advantage of this are realizing the benefits in a big way: according to the 2016 State of DevOps Report from Puppet, high-performing organizations deploy 200x more frequently, recover 24 times faster, have better employee loyalty, spend 22% less time on unplanned work, and 29% more time on innovating and building new product.

Organizations that want to excel in an increasingly competitive landscape are recognizing that their developers can no longer just throw their code over the wall to operations when it's ready for production. The new software development lifecycle doesn't just end at deploy. Rather, it looks something more like this:



Instead of turning a blind eye to code once it's out in productions, day-to-day performance monitoring, issue detection, maintenance and debugging is now a shared responsibility between development and operations (i.e. the NOC, help desk, etc.). And increasingly, managing code is becoming entirely the responsibility of the developer that shipped it.

Your development team is responsible for the customer experience. This is an awesome responsibility, and the best teams embrace it and thrive.

Why Developers Should Manage Their Code: Top 3 Benefits

Before we discuss best practices that help you streamline the process of having developers manage their code in production, let's spend a little more time exploring the benefits.

01 Increased employee engagement and morale

According to the Stack Overflow Developer Survey 2017 — which surveyed 64,000+ developers and gathered insights on demographics, top technologies, career satisfaction, and more — developers chose customer satisfaction as the most important and best way to evaluate the performance of themselves and their peers. Being on time and on budget came second and third in the list of metrics they felt were important in measuring performance. In other words, developers feel that they are doing a better job when they are closer to and can positively impact the customer experience.

Not only that, they want to impact the customer experience quickly and frequently. In the prior year, Stack Overflow's Developer Survey 2016 also found that there is a strong correlation between job satisfaction and number of times developers released code — a 77% satisfaction rate among developers who commit code multiple times per day, vs. 65% satisfaction rate among developers who don't check in code.

Developers are more creative and engaged when they work somewhere they feel is innovative, where they can rapidly and safely release directly to production. This ties into increased empowerment and accountability in fixing bugs and shipping code that is more production-ready, as they know their work is important. Having the power to delight end users is an amazing thing.

02 Improved application performance

Developers are the best candidates to help monitor and manage their code in production, for several reasons.

First, issues get fixed more quickly, with less friction. Developers can immediately roll back code that causes issues. And when minutes of application downtime can lead to thousands of dollars lost, it's essential to have the subject matter expert on hand that can get to the root of an issue and resolve it fastest (and also prevent it next time). What better subject matter expert is there than the person who built the service?

Secondly, developers that help run the production environment understand real workloads, and write better code when they feel the pain when it breaks. They are better at anticipating real scenarios for failure, as well as determining which system behaviors are related to common user problems that urgently need a solution. By having that understanding and visibility, they deliver more performant code that is production-ready. A continuous feedback loop and processes around constantly learning from operations data helps developers build services that are increasingly resilient, and over time they end up spending much less time fixing issues while on-call as their services are less likely to page out.

03 More time for innovation

Ultimately, developers who spend time on-call and become good at managing their code in production will get back more time to actually innovate. Several studies have shown that developers spend upwards of 50% of their time every day testing and fixing bugs — much of which may still escape into production — significantly cutting into time to write new software. On the other hand, fully replicating the production environment within pre-production and test isn't always possible. By having developers run the production environment, they'll have a deeper understanding of potential issues and be able to distinguish which are from code, and which are from infrastructure.

As such, developers that run the production environment will be able to ship cleaner, more performant code. By doing so, they'll spend less time on debugging as well as on fixing problems while on-call, and have more time to spend on doing the things they love (such as shipping new code, sleeping, or spending time with family).

A Case Study

Several years ago, a leading provider of cloud services for video began embracing a DevOps model to give their engineers more ownership across the entire software development lifecycle from design to management — so they could ship higher quality code faster. They also recognized that many DevOps transitions can fail due to a lack of cultural and technological change to support the shift. To tackle this, the engineering team took the first critical step by putting their developers on-call (specifically, on PagerDuty).

After putting their developers on-call, they were able to realize several benefits:

- Clearer visibility into the ownership and resolution of issues
- Reduction in the number of incidents as well as mean time to resolution by half, due to better collaboration and visibility between development and operations
- Reduction in on-call pain by minimizing time spent on administrative tasks and ensuring full resource coverage for all issues, with automated on-call scheduling and escalations

"I can't imagine life without PagerDuty."

– Senior Engineering Manager

5 Ways to Empower Your Team

Ultimately, the best way to empower your teams to do their best work is to get developers on-call for their code in production. A process that must be made as easy as possible.

Here are the key ways you can make going on-call seamless and meaningful for developers:

01

Help them get ahead of customer experience issues

The performance of an application or service is impacted by many factors — the external or internal cloud hosting environment, the network environment and other infrastructure, and services outside your team's control. In this context, having a comprehensive, live view of all data sources, including application performance monitoring data, network health, social media feeds, and more is essential. With a centralized visualization of all relevant data sources, developers can see clusters of events across tools that don't necessarily talk to one another, that can reveal the customer impact of a deploy gone bad or indicate simmering issues that inform critical development decisions.

02

Ensure they can focus on what matters

Developers don't want to waste time trying to tune filters and thresholds in upstream monitoring tools that they may not even have access to. As such, all alerts should be centralized in a single location, in which they can define simple rules for controlling alerting behavior and workflows — including the grouping of alerts that are related to the same issue and suppression of any non-actionable alerts.

Instead of getting an explosion of email notifications every time a service is degraded, ensure developers control the means to only get woken up when it's something that warrants their attention. When they are paged, all the related contextual information should be grouped into the same incident object so they don't need to manually deal with thousands of one-off alerts. Give them the means to automatically cut through the noise so they can focus on what matters.

03

Equip them to automate and reduce manual toil

Implement a standardized process around incident response and training, and ensure that developers can automate repeatable tasks. The response process ideally should be centralized in a single location so developers can quickly remediate issues without being burdened or stressed out by needing to context switch or waste time trying to find the right information.

By putting steps in place to automatically surface real-time monitoring iframes and metrics within incidents, for example, you give developers the ability to minimize the cognitive load of understanding the actual scope of an issue. Even better, developers should be able to try simple fixes and troubleshooting actions (i.e. "restart server") without needing to manually switch into other tools.

04

Choose solutions that are extensible for them to build desired workflows

Invest in solutions with powerful, extensible APIs — that way, developers can optimize existing toolchains and work their way. Ideally, certain actions can be automated through the API to facilitate simple, powerful end-to-end incident resolution workflows, such as creating and resolving incidents, merging incidents, normalizing inbound event data, and more.

ChatOps is also another way in which developers can build software and fix issues in a collaborative and powerful way. Tools that support ChatOps integrations enable developers to fix issues in-line and automate ops-related tasks with slash commands and chatbots — this also helps minimize context switching.

05

Enable developers to continuously learn and improve

Last but not least, it's essential that developers are equipped with the means to continuously learn and improve. Many teams perform post-mortems to extract learnings from previous outages. However, this is often a highly arduous task, with team members spending hours for every incident retroactively writing up and piecing together information from silo-ed locations.

Investing in a solution that streamlines post-mortem creation by automatically pulling in the relevant incident context such as timelines and root cause, is essential. Teams should also have access to analytics and metrics around system and team operational efficiency, such as mean time to acknowledge and resolve. These safeguards minimize friction in the process of learning from mistakes and extracting actionable insights, so that your team can constantly improve together.

Being on-call can be a big source of empowerment, not stress. By making on-call as painless as possible through implementing the best practices outlined above, your team will be positioned to do the most impactful work of their careers — directly driving the success of your customers and your organization.

Try PagerDuty Free for 14 Days