LaunchDarkly

10 Critical KPIs for Stronger Software Releases

The best metrics to maximize your team's quality and efficiency.

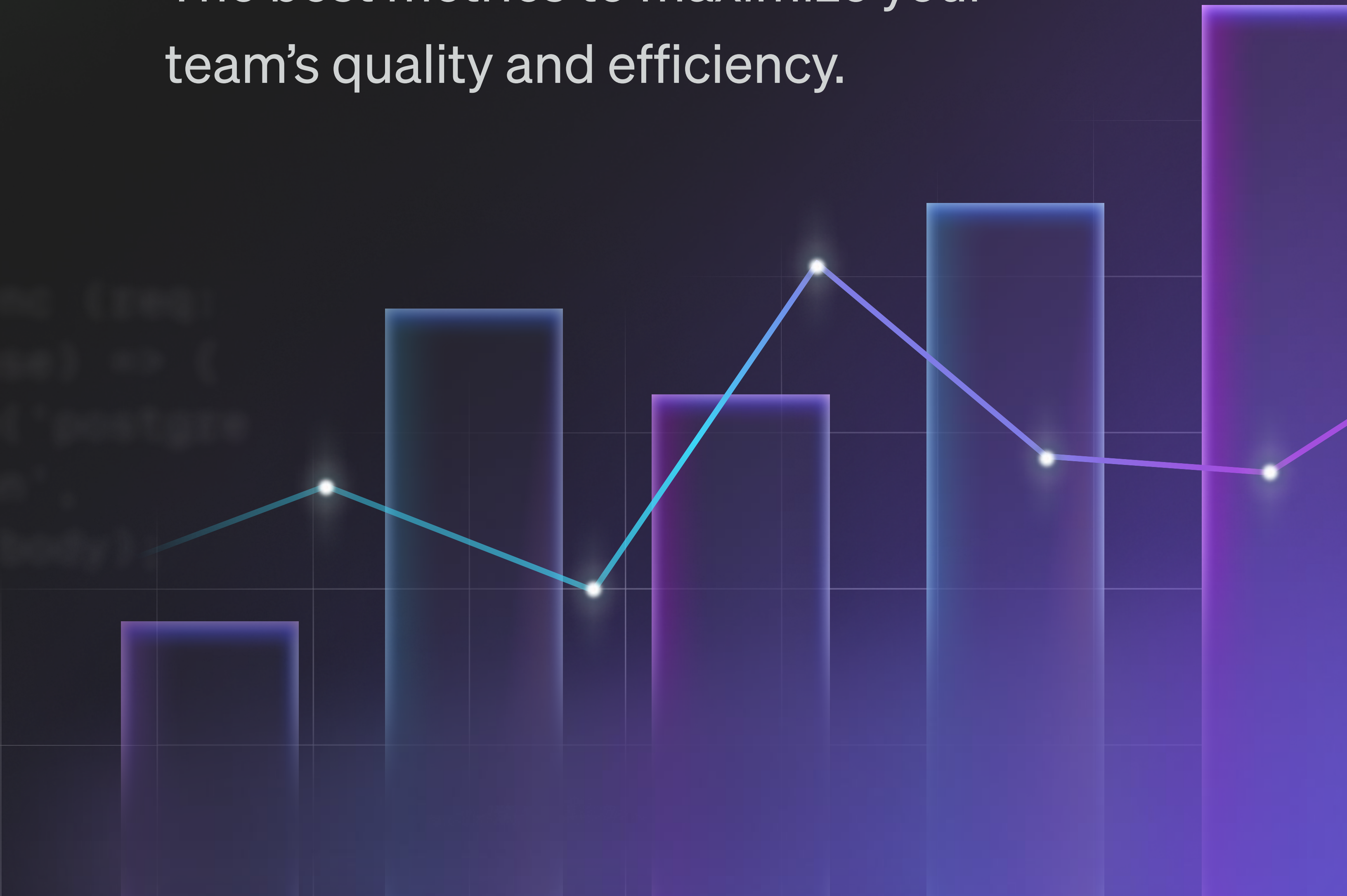


Table of Contents

INTRODUCTION

The Value of Better KPIs

Key performance indicators, or KPIs, are defined benchmarks to understand how your organization is performing. At many companies, KPIs are used by nearly every department, from human relations to marketing.

In software development, KPIs can take on many familiar and quantifiable forms, such as the number of monthly active users or the amount of revenue your app brings in every quarter. KPIs are important because they help your teams determine how successful they are at achieving their goals.

In this guide, we'll suggest the 10 KPIs you can use to track how well you're deploying, releasing, and shipping software. Although these are just our suggestions, we believe they represent a good starting point for data collection during an often overlooked phase of software development. That said, one size doesn't fit all teams, so you may want to pick and choose.

01

Release Frequency

Back in the day, when software was shipped in compact discs (and before that, floppy disks), the concept of a release was simple. You typically release one major software update every year or two and then spend the rest of the time preparing for the next year's cycle.

Even though almost every app is distributed through the Internet, major releases continue to be few and far between. Instead, the year is punctuated by minor and patch releases, which consist of anything from new features to essential security updates.

Being able to release software quickly and without obstacles is a good practice. But tracking the frequency of these releases can also set a baseline gauge for overall productivity.

For example, if you have sprints that last two weeks, you could track how many of those end with shippable code. This can help you establish a more appropriate cadence for sprint lengths. It might also help you discover how your distributed teams are functioning together.

02

Feature Time-to- Production

Another metric to capture is the entire life cycle of a feature, from the time a developer accepts the work to when it appears live in production. Capturing how long it takes for an idea to become reality can help uncover issues such as resource misallocation or unrealistically tight deadlines.
If you're noticing that features are taking longer to complete than expected, it could mean that the length of your sprints or your completion time estimates are optimistically unrealistic.

Supercharge Your Release Pipeline

As organizations grow, it's tempting to stick with the way things have always been done. And yet, stepping back and asking questions can yield big results. In this session from LaunchDarkly's recent user conference, Galaxy, Caley Brock, a senior software engineer at VTS, talks about some of the improvements that changed her team's process and enabled faster shipping and more streamlined testing.

Watch now →

03

Deployment Speed

KPIs rarely function in isolation, and trying to answer even just one question may involve looking at more disparate data points.

Timing the speed of your deployments —as well as other indicators like its 99th percentile maximums—will let you know concretely how long it takes to ship a new change. This includes timing your infrastructure (how long it takes for code to be deployed, how long it takes for servers to reboot) as well as any CI (continuous integration) test suites that must return green before a deployment can proceed.

If you observe that teams are releasing infrequently, the length of time it takes for a deployment to succeed might be one reason why. This hard data is an opportunity for determining how to speed up the overall process. Deployment speed also allows for faster feedback cycles to determine if the feature is useful and operating correctly.

04

Release Downtime

Few modern applications consist of just a single monolith. Additional infrastructure, such as databases, key-value stores, and event queues are often necessary data storage components.

If these systems are paused, or even “harmlessly” error out during deployments, it could be indicative of larger structural problems in your processes.

Plan for unexpected downtime and add mitigations such as retry queues or other fail-over tactics. If these other systems are interrupted for extended periods of time on every deployment (or there are a burst of errors on every release), it could mean that you ought to add more fail-over mechanisms, tests, or looser coupling.

Launch with Reduced Risks

If you’re skeptical of how shipping faster can reduce risk, check out this presentation from LaunchDarkly’s Principal Developer Advocate, Heidi Waterhouse.

Watch now →

05

Number of Expected Deployments

Typically, you’d want your software releases to occur on a planned schedule, such as after a sprint or on a specific day every week. This helps plan work around gathering feedback and keeps work flows consistent. But of course, in reality, you may be releasing several times a week, to patch bugs and include other hotfixes.

A steady increase in the number of hotfixes could be a sign that more emphasis is needed for your team's QA/testing stages, or that you have a lot of technical debt in your codebase. Focusing on testing, code cleanup, and stability could mitigate the frequent problems cropping up in production. Identifying how many releases correct issues in earlier versions could help you identify where in your feature life cycle more cautious code review is necessary.

06

Number of Rollbacks

Of course, as developers, we've come to expect the unexpected when code and users meet. Unplanned situations are rare, but they do happen; if they're dangerous enough, it might necessitate a full reversion of newly- deployed code.

You can get ahead of these sorts of problems by tracking the number of times your teams perform a rollback. Don't think of a rollback as a mistake made, but as an opportunity to strengthen the development and review stages that always occur before the deployment phase.

Ways to Measure Software Quality

There are a set of measurable leading indicators to achieve desirable outcomes in software quality. In this presentation, Stephen Magill, VP of Product Innovation at Sonatype, explains the most effective measures of software development and operations processes.

Watch now →

07 Test Speed

Since testing is such an integral part of releasing software, it too can surface KPIs for tracking. One quick way to do this is to track how long it takes for a test suite to finish executing. If your suite takes half an hour, it might be responsible for infrequent deployments or otherwise hinder productivity. Long test times could be an opportunity to introduce more static fixtures or parallel execution to speed things up.

What Is Continuous Testing?

Continuous testing provides an automated, end-to-end testing solution that can integrate with an existing development process. It can eliminate errors and make true continuity easier throughout the development life cycle. In this article, we discuss the various ways your organization can benefit from implementing continuous testing.

Read now →

08 Test Coverage

We all know the value of good tests, but how can we be sure that our tests are, in fact, good?

Quantifying test coverage as a KPI can ensure that at least most aspects of your app are being thoroughly tested.

Release Testing Explained

Release testing refers to coding practices and test strategies that give teams confidence that a software release candidate is ready for users. Release testing aims to find and eliminate errors and bugs from a software release so that it can be released to users. In this article, dive in and explore several methods used to perform release testing.

Read now →

09

Test Flakiness

All too common (and frustrating) in software testing is the appearance of the dreaded test flake.

If you're unfamiliar with the concept, a flake occurs when an individual test fails on one test run, but then passes on the subsequent test run. Many testing frameworks run your tests in a random order precisely to surface test flakes, as proper test-passing software should not depend on the order of the test.

The appearance of a test flake might be a one-off, or it may indicate a less-than-ideal testing approach. Worse, it might even indicate a significant bug in your software.

Whatever the cause, flaky tests undermine confidence in the entire test suite, not to mention extend the deployment time since the test suite needs to be re-run to pass.

Your test-flake count should always be at (or at least, trending down towards) zero.

10

Feature Lifespan

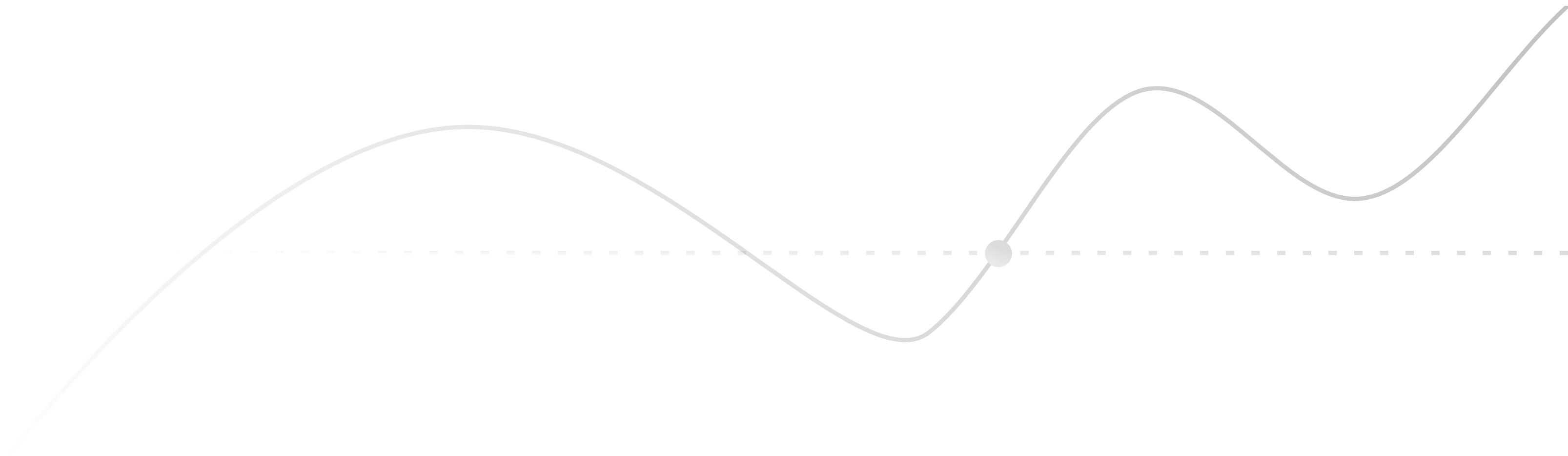
Once a feature is live, start measuring how long it remains a part of the app against how often it's used. For example, you could consider eliminating a feature added four years ago but used by less than 1% of your user base.

Removing unused code from your application is about more than just keeping things clean; it's also a way to keep technical debt from piling up. Sometimes, more modern features are inextricably dependent on outdated systems. By consistently updating older features—or better, pruning them completely if they're mostly unused—you reduce the effort needed to make future progress.

Wrapping Up

Choosing what to measure for your KPI is only one part of the equation: it’s up to you to determine what constitutes a “good” metric. If your tests really do need to take half an hour to run, then your numbers will clearly be different than an app whose test suite takes thirty seconds. Nonetheless, there will still be the same concerns around keeping to that predefined baseline.

Once you’ve figured out what to track, there’s the question of how to track it. You’ll likely be able to integrate into your existing monitoring software, whether that’s Prometheus, Grafana, or Datadog.
 And, although these are just some of our suggestions, you’re likely to come up with many more KPIs that are more relevant to your specific needs. Your choice of KPIs might be subjective, but KPIs are critical to producing quicker, consistent, and error-free software releases.



Empowering all teams to deliver and control their software.

Get a demo

launchdarkly.com | sales@launchdarkly.com