split

Feature Flags Are Changing Observability As You Know It

Even the best written code breaks sometimes. Even the most well-built features can have unforeseen downstream effects. These challenges are an inevitable part of the development process, but to keep your software healthy and customers happy, fast and effective fixes are key.

Of course, there's no incident response without visibility. You've probably implemented monitoring tools to report on your system, but how do you get from knowing what healthy performance looks like to quickly detecting and dealing with issues, all without **false positives?**

Observability: The Three Pillars

Put simply, observability is the ability to understand how your entire system works and fits together. From knowing how everything works, you can identify the performance factors to monitor and pinpoint problems in the internal system based on the external outputs you get. In other words, you can navigate from the effect to the cause.

It's broadly accepted that there are three pillars of observability:

Metrics

Numerical values related to your system. Tools that collate, analyze, and visualize metrics allow you to understand how your system performance fares over time and across infrastructure components.

Logs

Records of actions and events that happen in a system, including fault details. Logs help you troubleshoot code, identify where and why errors happen, and identify unpredictable and emergent behaviors.

Tracing

Following the whole journey of a request or action through your system. Trace data lets you profile system health and helps you identify the most valuable logs and metrics to use for analysis and troubleshooting.

Having all these capabilities certainly helps your system become more observable — but as your software and infrastructure gain components and complexity, you'll find that the devil really is in the details. For instance, while you're already using a comprehensive tool that offers metrics, logs, and tracing (e.g., Datadog, New Relic, Dynatrace), it's likely that you're also still performing hotfixes and rollbacks because you're unable to pinpoint the root case of code issues.

Rollbacks and Hotfixes: Problems of Scale

Have you ever shipped a feature to end users, only for performance issues to flare up somewhere in your system? It's a high-stakes, high-stress situation; you've got to figure out what exactly is causing errors and decide whether to roll that feature back or hotfix it, all while managing other parts of the development process.

For troubleshooting performance, you want observability on your side. As your software grows, the possible failure nodes multiply along with it — and with the pressure for teams to develop and deliver faster, identifying where and why problems arise in your system is an increasingly difficult task.

When rapid feature delivery meets a lack of visibility, this leads to more unchecked failures and, in turn, more emergency recovery measures that can bring further risk and inefficiency.

Split's **research** shows:

- **67%** of organizations release features on a bi-weekly basis
- **27%** of organizations experience downtime after new releases
- **41%** of organizations must either roll back or hotfix more than 10% of new features
- **38%** of organizations take more than a day to detect issues with new releases

Here's the bottom line: companies are releasing features quickly, but they're frequently running into issues with system performance and timely error detection. And in these situations, you don't want to rely on a rollback or hotfix to save the day.

Rollbacks rarely provide fast solutions. If something goes wrong and developers can't identify which particular change caused it, they need to roll back whole sets of changes to correct the error — a process that can prove disruptive.

Meanwhile, **hotfixes** up the risk factor significantly. Rather than reverting a change, hotfixes see developers push code changes live into production without thorough testing, in hopes that they'll fix errors — a gamble that often introduces entirely new problems.

If you can't quickly detect issues and fix them reliably, your user experience suffers. And that's a bad sign for your company's bottom line and reputation. The good news? As software has evolved, so has observability.

Feature Flags: An Added Dimension

In modern software development, monitoring and observability must occur at the individual feature level. Why? Because when you track granular changes, you see the root cause behind each new effect — and that's essential to stay on top of error detection and response.

Let's say your new release is a mixed success: it hits the goal you were aiming for but negatively impacts another element of the user experience. When you're observable down to the level of each feature, you don't need to make surgical rollbacks or cross your fingers for a hotfix that works; you can just stop the feature immediately, solving the user experience problem, and apply a fix or refinement on your own timeline.

You do this by using **feature flags** — mechanisms that let you choose between different code paths in your system at runtime. Often, feature flags look like "if/else" statements in code that you use to toggle releases and experiments on and off. Feature flags can be combined with metrics data to provide a view of each feature in isolation. From this, you can understand the impact of each feature and act quickly to address it.

In that sense, feature flags give you visibility into where things happen and control over who features reach and how they're rolled out. These are both essential ingredients to experimentation and quality assurance, letting you limit the blast radius when an error happens, or an idea doesn't perform the way you want.

Observable Features: The Next Level

While there are monitoring and observability tools out there to cover the three pillars, Split's feature delivery platform complements them by combining feature flags with data analytics to make your overall observability posture even stronger.

Being observable leads to being proactive — and automation is crucial to this. Data on dashboards is just the starting point for what monitoring needs to do for you; fast releases, fast troubleshooting, fast iterations are all near-impossible without an automated engine that alerts you of the business impact of each change with analysis of the root cause.

If your team needs to dig through the system to find this information, then that's just another manual task in the way. This is why Split's platform comes with **Feature Monitoring** — an automated detection function that alerts teams to errors and performance issues in code releases. It goes beyond traditional monitoring by connecting issues to specific features, making them genuinely observable.

By isolating the impact of each feature, Feature Monitoring lets teams deliver on core response metrics:

- **Reduce time to remediate** iwith alerts that tell you exactly where the problem is and feature flags that let you shut down errors with a click before they detract from the user experience.
- **Impact fewer users** by making releases available to only small segments of your user base.
- **Speed up time to detect** with an analytics engine that measures performance and detects issues in real time.
- **Eliminate alarm tuning** by treating feature releases as randomized controlled trials. This accounts for gradual system changes, other releases, and external factors like marketing pushes; the alerts you'll get for each feature are strictly relevant to its performance.
- **Cut down on false alarms** by using performance statistics to drive incident response.

Split's feature delivery platform brings feature flags, automation, and analytics together to speed up detection and response, giving you the visibility and insight for fast and effective experimentation.

Want to test an idea? Split integrates feature flags with the data you need, ingesting metrics from any source, automatically calculating the impact of each feature, and quickly finding the root cause. You'll know exactly how customers respond to your features — and with that info, you can make them more effective with each iteration. Aim higher, achieve more.

Split in Action: A Success Story

Surfline, the world's largest surf forecasting site, partnered with Split to improve its development velocity and scale its analytics across all feature releases. Here's what they achieved:

Greater Focus on Innovation
Before Split, rollbacks could take from a couple of hours to a day, and with Surfline deploying twice a week, the equivalent lost time was half that of a full-time engineer. By using feature flags to decouple deploy and release, Surfline gained back this time and can focus on making improvements.

Better Data Analysis
Surfline used various data analysis tools but found it challenging to measure performance against KPIs for every feature release or in testing and staging environments. With Split's analytics integrations and experimentation support, Surfline now analyzes the results of feature rollouts and tests across all of their metrics.

Surfline now has the observability to view each release as an opportunity to experiment and refine features to be their best.

Together, feature flags and automated analytics push your observability efforts forward. By seeing how each individual feature behaves and how it measures up to your business outcomes — layered with the power to toggle it on and off — you get the speed, visibility, and control to fix and refine your features. This means you can make your features the best they can be and release them at the pace you want.

Observability at this level opens up a few new doors for your team. For one, you can achieve progressive delivery — the ability to implement gradual feature rollouts at any time, using feature flags to quickly test, release, and neutralize errors. It also helps to create a culture of knowing better and doing better; your team is more engaged and responsible when it knows where and how to improve, and that there's always room to try out the next big idea.

Still dealing with rollbacks, hotfixes, and unknowns in your system? Split gives you a fast and easy way to control risks and gain performance insight. ***Get in touch*** *to see what feature flags and data can do for you.*