Reconciling change management and continuous delivery.

The secret to automating releases, mitigating risk, and maintaining compliance.

LaunchDarkly

Contents

Introduction

Software and innovation go hand in hand. Indeed, what business today can profess a need to innovate faster without mentioning software in the same breath? In 2015, McKinsey warned that "new research suggests that companies pay a price when they undervalue the strategic importance of producing excellent software." Subsequent research agrees.

To furnish excellent software, you must excel at delivering software. The DevOps Research and Assessment (DORA) group offers a helpful model for measuring software delivery performance. In broad terms, they look at speed, stability, and reliability.

Some organizations have it all: high speed, high stability, and high reliability. But countless others can only lay claim to one or two of those performance markers. And when the choice becomes either speed or risk mitigation, the latter often prevails. Erring on the side of caution is wise, of course, when dealing with things like personal health data, financial assets, and national security. Even so, if software development is critical to your business, then speed must be critical as well.

Unfortunately, many companies are hampered by sluggish bureaucracies. They run new software features through heavy change management protocols such as those involving a change advisory board (CAB).

Gartner predicts that "by 2023, 90% of organizations will fail to meet required release velocities to deliver full business value to their customers if they use a centralized release management approach in DevOps environments." That's a mouthful. But if Gartner's prediction contains enough truth, then the question organizations need to answer is:

How can we accelerate software delivery—i.e., innovate faster—while mitigating risk and maintaining regulatory compliance?

In this paper, we'll explain how feature management answers that question. Further, we'll show how to leverage feature flags within a feature management platform to:

- Decentralize software change approvals
- Cut out the inefficiencies of change management while amplifying its positives
- Enable safe, compliant continuous delivery
- Automate release workflows in which change controls are embedded
- Advance platform standardization

In short, we'll explain how you can have it all.

Why heavy change management exists in the first place

Traditional change management frameworks, such as the popular Information Technology Infrastructure Library (ITIL), serve an important purpose. We're not here to denigrate ITIL.

It's worth noting that ITIL, too, has evolved to meet the demands of a fast-paced digital world. For example, a guiding principle of ITIL 4 is: progress iteratively with feedback. This aligns with modern development ideas like Agile, DevOps, and continuous delivery.

Here are some of the pros of legacy change management:

1. For one thing, it governs far more than just deployment and release management. Software updates represent a fraction of all changes that a CAB oversees.

2. It can mitigate risk and guard operational health in some circumstances.

3. It ensures regulatory compliance. For example, we interviewed a publicly-traded U.S. company that supplies electronic health record technology, and they attested that their CAB helps them comply with government regulations such as SOC II.

4. It supports auditability. In the absence of modern tools that automatically record application updates (e.g., version control systems), change managers help log software changes. This paves an audit trail for regulators. Such tracking also preserves institutional knowledge.

5. Change management helps maintain harmony between a multitude of engineering teams, informing each of what the others are doing.

The downsides of heavy change management

More often than not, technology leaders, developers, and even IT operators recognize the limitations of change management. Here are a few such drawbacks in the context of software delivery.

Change management slows your time-to-market

Every day you build, test, and deploy a feature, you incur overhead. If customers detest or ignore what you've built, then you've wasted valuable resources on a fruitless endeavor. You've also incurred opportunity costs. In the spirit of ITIL 4, we advise shipping small changes, ideally to a production environment, as early and often as you can. By gathering real user feedback over a short interval, you can more rapidly decide whether to iterate on or discard a feature.

Studies show that bureaucratic change approvals are bad for business

In the State of DevOps 2019 Report, DORA authors found that heavyweight change approval processes were associated with worse software delivery and operational (SDO) performance. According to this same research, high SDO performers were twice as likely as low performers to meet or exceed key business targets: profitability, productivity, and customer satisfaction. Not only that, heavy change management was correlated with worse change failure rates. That is, it actually *increased* risk.

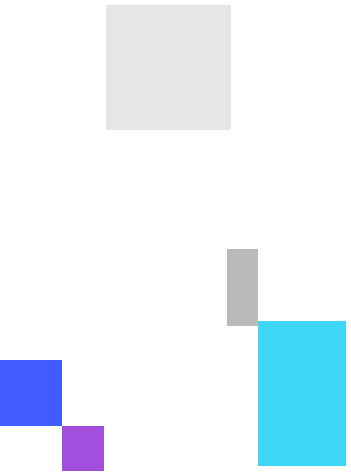
Other researchers have drawn similar conclusions. Forrester, as an example, published a 2020 report based on surveys of hundreds of IT operators and found that "organizations who have taken action to streamline their change process were much more likely to report superior business performance."

Developers generally dislike CABs and RFCs

Development teams often see bureaucratic change management as a chronic frustration. For example, the Director of Enterprise Engineering at a Fortune 500 financial services company, a LaunchDarkly customer, expressed the following in an interview:

> **Change management is a big burden on our digital product owners and developers. No one likes it.**

If your most talented developers endure inefficient processes long enough, their passion and productivity will wane. Many will leave. Failure to attract and retain talent puts your business at risk.



How feature management accelerates software delivery and streamlines change management

Feature management is a class of modern software delivery tools and techniques anchored in feature flags. LaunchDarkly is the industry's first feature management platform.

We use LaunchDarkly to release new features to a subset of our guests, gather responses, fine-tune our approach, and validate new features in production to be able to deliver a better experience for all travelers.

Manjari Ranganathan
IT Manager
Hawaiian Airlines

Three core capabilities underpin feature management.

1 Decouple deployments from releases

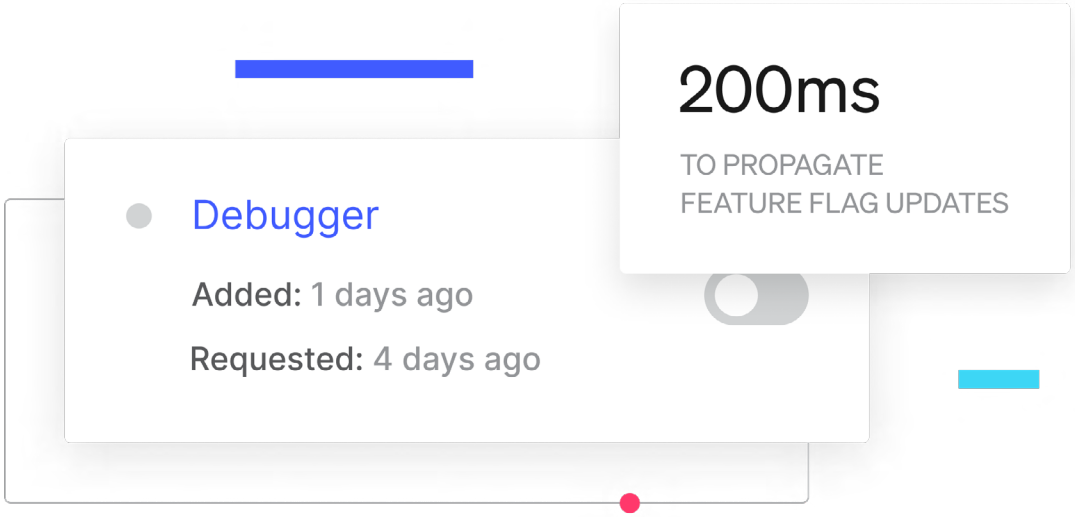
In basic terms, “deploy” refers to moving a code artifact into a production environment. “Release” refers to making that code available to external audiences. Historically, businesses have been unable to sever *deploy* from *release*. Thus, whenever they deployed code, they had no choice but to release it to all users at the same time. The risks of such an arrangement are high.

Feature flags, on the other hand, let you separate code deployments from feature releases. This enables developers to deploy without fear. Moreover, it allays some of the concerns IT operators have about code being shipped to production frequently. And it lets product managers and release managers control the pace at which customers receive new features.

Of note, in its list of 34 practices supporting the “service value chain”, ITIL 4 puts “deployment management” and “release management” into separate categories.

Fail-backs with LaunchDarkly have been a lifesaver for us.

Associate VP of Engineering and DevOps Fortune 500 health insurance company



2 Instantly roll back with kill switches

Another chief benefit of feature flags is they allow you to disable broken code at **runtime.** That is, you can remotely turn a feature on or off in production without having to push code through your deployment pipeline—or change approval process. When an incident occurs, you can toggle the flag associated with the error and fix the problem in milliseconds. The stress, engineering costs, and bad customer experience all but evaporate.

An Associate VP of Engineering and DevOps at a Fortune 500 health insurance company declared that such “fail-backs” with LaunchDarkly have been a “lifesaver” for his team.

Lead time for changes

A 2020 survey shows that, after using LaunchDarkly, customers saw a

76%

decrease in their lead time for changes.

3 Employ fine-grained targeting

Feature management supports targeting across discrete users, environments, geographies, customer sites, and other endpoints. You can create segments based on a host of custom attributes. What's more, you can leverage percentage rollouts and targeting in tandem. Such capabilities give you tremendous control over how, when, and to whom you ship features. In keeping with ITIL 4, they allow you to progressively deliver new functionality.

LaunchDarkly customers have used the three core capabilities of feature management to improve their software delivery performance:

Deployment and release frequency

Northern Trust went from 2-3 deployments per month to 21 deployments.

Mean time to restore service (MTTR)

It used to take Paramount up to a week to fix bugs, but now it only takes a day.

Change fail rate

LaunchDarkly helps Honeycomb maintain a 0.1% change fail rate.

Feature management provides the safeguards to decentralize software change approvals

With feature flags in the picture, IT operators and change managers can feel confident about designating more software changes as “standard”. In the ITIL model, the development team is permitted to approve “standard” changes through peer review. Delegating approvals benefits everyone. Developers avoid delays. Operators get peace of mind about risk mitigation. And relations between the two improve.

Let’s illustrate how this might look in practice.

Imagine you have a developer who wants to deploy a feature to production. First, they wrap the code in a flag, thus barring users from activating the code pathways. This represents the first layer of risk mitigation.

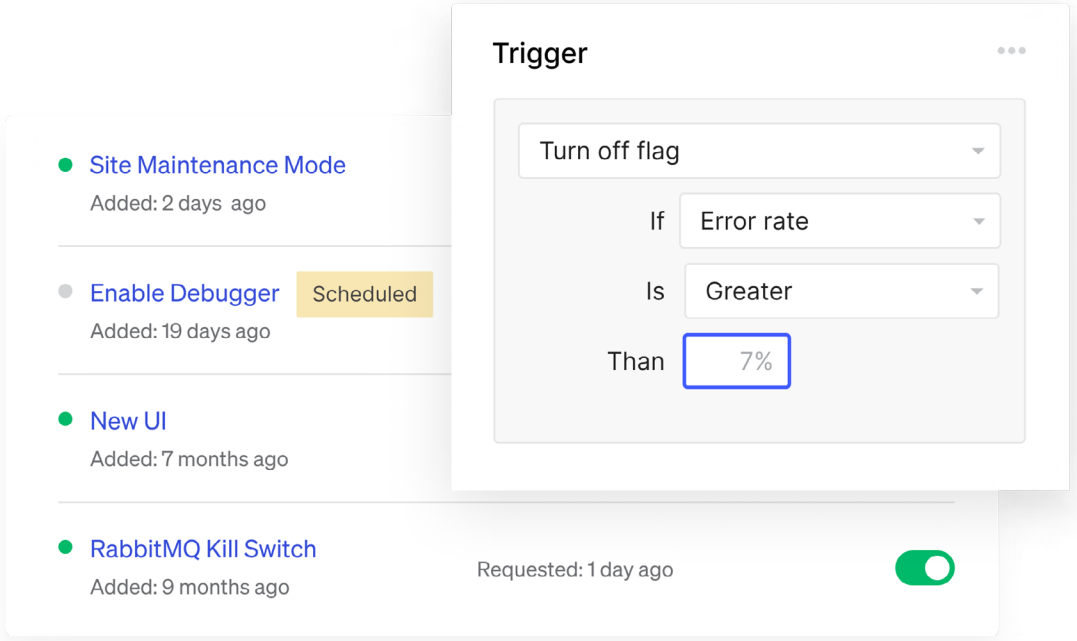
If for whatever reason the feature causes a bug, one person—a developer, digital product owner, IT service operator, SRE, etc.—can turn it off immediately. No prolonged rollback. No fix forward. No emergency code change that must traverse the whole deployment pipeline. The feature flag kill switch thus constitutes a second layer of risk mitigation.

..we sometimes need to ship changes to [our application] at a faster pace than is ideal. In the past, we'd have to run these changes through a rigorous approval process before going live. It was both slow and stressful. But now our attitude is: 'We trust what we ship, but we always have a backup plan wit LaunchDarkly.'

Artie Lee
Director of Engineering
Climate LLC, a Bayer company

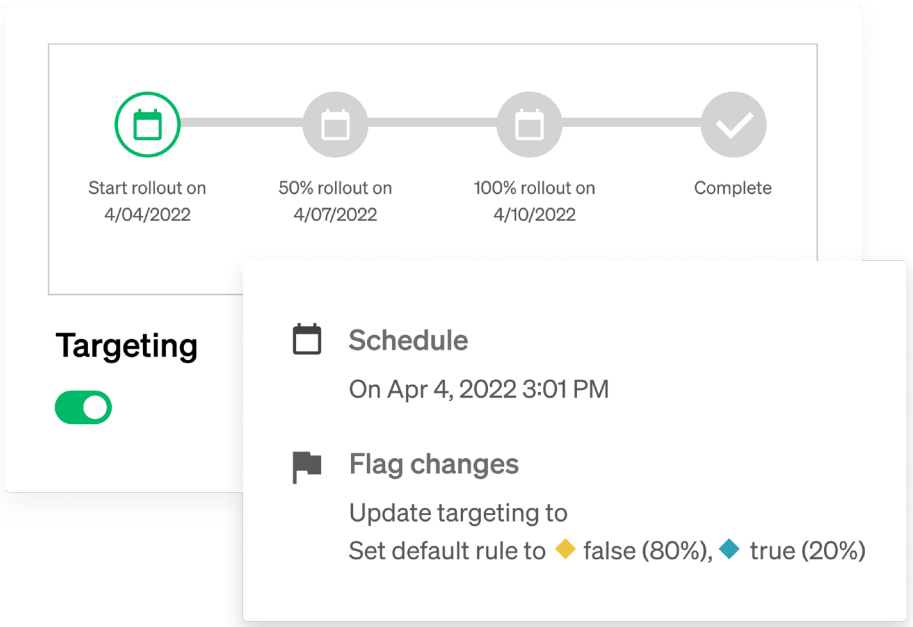
With Flag Triggers, you can resolve the incident in near real-time. Our platform integrates with popular observability and application performance monitoring (APM) solutions. In cases where error rates exceed a certain threshold, your APM will trigger LaunchDarkly to disable the feature flag associated with the errors. This happens *automatically.* And it represents yet a third layer of risk mitigation.

You also mitigate risk through targeting. For example, you can safely run canary tests of new features with a small subset of users in production. If something goes wrong, you've limited the blast radius to just the canary group.



Feature management embeds change management into automated software release workflows

Feature flags alone can take you a long way toward enabling safe, compliant CI/CD. But at LaunchDarkly, we've taken this a step further by automating key parts of the software delivery process. Through Workflow Builder, a part of our Feature Workflows offering, we let you embed change controls, compliance requirements, and quality gates within automated release workflows. Moreover, we let you create custom release templates that you can use repeatedly.



Here's what this looks like in practice.

1. One of your developers builds a substantial feature and wraps it in a LaunchDarkly flag. This change requires approval before production.

2. The developer uses Workflow Builder to assign a change approver from among the engineering managers.

3. The developer schedules a future date, time, and target segment for the initial rollout. In this case, your team plans to progressively deliver the feature to six different segments, ending with all users and environments (aka a "ring deployment").

Each successive segment, or ring, carries more risk. As such, you might make the approval requirements lighter in the early stages of the release and stricter in the later stages. For instance, if Stage 1 entails deploying to just internal developers, then you may decide to forgo approvals. But if in Stage 5, you're shipping to a geography where your application has previously malfunctioned, then you might require an engineering leader, senior IT operator, or even a CAB to sign off on the change.

In any event, you can easily set all such custom rules with Worfklow Builder.

We tied LaunchDarkly to ServiceNow to kick off a change control ticket every time a flag is flipped, so that each toggle is logged. Automating the creation and closure of change control tickets was a big factor for teams to be able to track their changes, since they normally would require a full intake process and technical review from our enterprise change management team and would have to schedule release windows.

Nicholas Goss
Architecture Senior Designer
Cigna

4 LaunchDarkly automatically initiates Stage 1 of the rollout, deploying the feature to production while only releasing it to internal developers.

5 In this case, let's assume the team is using LaunchDarkly's integration with ServiceNow, a top IT service management cloud software provider. As a result, when Stage 1 is initiated, it automatically generates a ServiceNow ticket.

6 In Stage 2, which again has been scheduled in advance, the feature gets released to a small beta group for testing. Developers, IT operators, and product managers monitor the situation.

7 Once the rollout reaches Stage 4, the assigned reviewer will approve the software change, thus, in part, meeting the SOC II "segregation of duties" requirement. The release proceeds as scheduled.

8 If your APM detects an error tied to the feature, it will automatically notify LaunchDarkly, in turn, disabling the feature. Whatever impact the error may have had on the user experience has been greatly diminished.

9 Once the feature clears every checkpoint and performs as intended, it is rolled out to all audiences. Then you become multi-millionaires. Obviously.

The benefits of automated release workflows

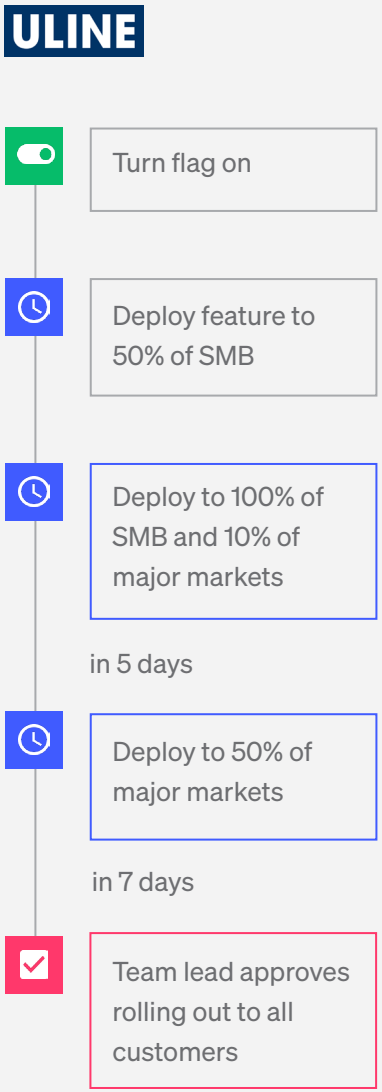
1. Workflows cut down on change management bottlenecks and augment the CAB's strategic qualities. Change managers can tackle other issues beyond the walls of software development. What's more, they can better serve as an information beacon that keeps developers across the company aligned. And, frankly, they're liberated from having to review every infinitesimal software change.

2. Workflows reduce inter-team dependencies and boost developer productivity. In fact, the LaunchDarkly safety net gives organizations the confidence to practice continuous delivery.

3. Workflows nearly eliminate risk from deployments and releases. If a feature causes a bug in production, you can disable it immediately. Moreover, with targeting and percentage rollouts, you can tread cautiously when shipping features.

4. Workflows help ensure regulatory compliance. And they do so while reducing toil and paperwork.

5. Workflows improve collaboration, especially between change and release managers. Traditionally, a clear division lies between change management



(a governance process) and release management (a technical implementation and operations process). But when you weave governance into software delivery, you inevitably draw change and release managers into closer alignment.

Here's how one LaunchDarkly customer benefits from Feature Workflows.

ULINE, a leading distributor of shipping, industrial, and packaging materials to businesses throughout North America, uses Feature Workflows for major digital product releases. They'll start by, say, releasing a feature to 50% of their SMB customers. Then, after a few days of monitoring, they'll expand the release to 100% of the SMB segment and 10% of major markets. This gradual rollout continues until, at last, a team lead approves shipping the changes to all customers.

> **We're able to coordinate multi-stage rollouts to automatically release changes to our customers without having to compromise between speed and operational excellence.**
>
> Aaron Jeske | Senior Software Developer, ULINE

Feature management advances platform standardization

More and more enterprises are building internal “platforms” to support safe, efficient software development, delivery, and operations. Here are a few ways in which LaunchDarkly meets the requirements of virtually any platform initiative. And here’s why we encourage platform engineering teams to use LaunchDarkly for standardizing deployments and releases.

Integrate feature flags with developers’ daily lives

API-first

Every feature in LaunchDarkly is API-first, giving you the flexibility to plug feature flags into your platform. Build integrations, export raw data, and write custom scripts for automating feature flag workflows.

SDKs and integrations

LaunchDarkly offers 25+ client-side and server-side SDKs tailored to every major platform. We also integrate with numerous tools you use every day for monitoring, collaboration, data analysis, and beyond.

Control access to LaunchDarkly based on your business needs

Custom Roles

Custom Roles give you control over who can access what in LaunchDarkly. This applies to everything from feature flags and Projects to Environments and Teams. Enforce access policies that meet your exact workflow needs.

For example, only permit a Director of IT Operations to make certain changes to production. Or, allow an engineering manager to alter flags in one microservice while barring them from another service area.

Teams

A Team is simply a group of members within your LaunchDarkly account. An account administrator can grant specific permissions to specific Teams. This makes creating and enforcing access rules more secure, scalable, and efficient.

Track every software change with an audit log

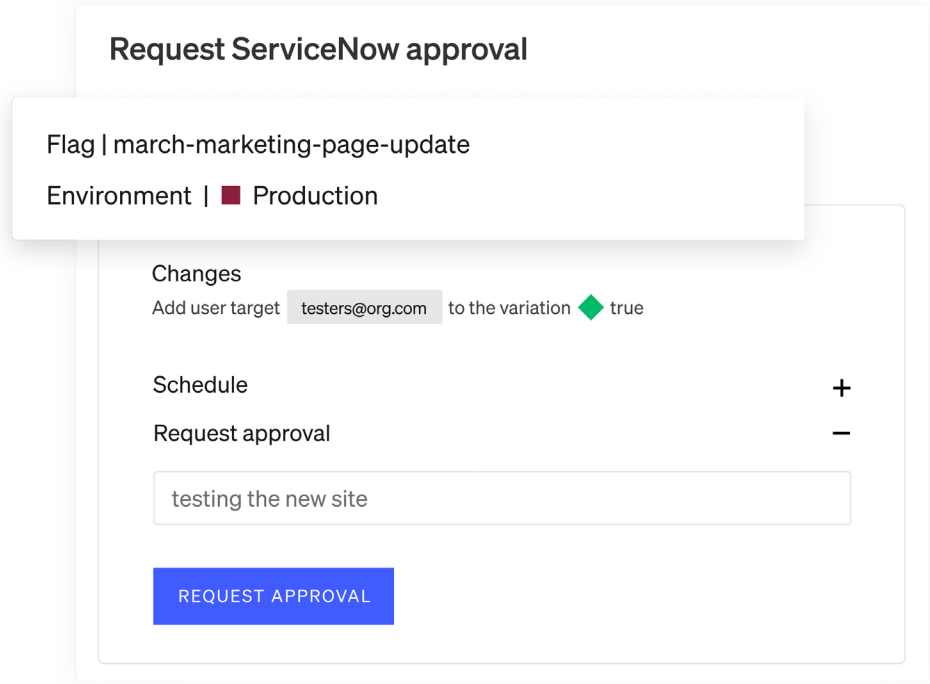
Audit Log

LaunchDarkly provides a full audit log that automatically records all software changes

you make involving feature flags. Change managers can present this log to regulators during an audit.

LaunchDarkly's integrations with Slack and Microsoft Teams also help you maintain compliance from an auditability perspective. You can view, monitor, and control feature flags within your Slack and Microsoft Teams workspaces. Lastly, you can also log changes automatically via LaunchDarkly's ServiceNow integration.

All of these automated change logs further support enterprise governance and compliance.



LaunchDarkly has enabled us to reduce technical debt and move to modern systems more quickly than we could have imagined.

Microservices Domain Architect
Fortune 500 energy company

Effortlessly manage technical debt

Code References

Code References in LaunchDarkly allow you to quickly find and remove stale flags from your code. At the same time, you can set up alerts for Slack or Microsoft Teams in which you get reminded to archive old flags. This helps you maintain impeccable code hygiene when using feature flags on a large scale.

With LaunchDarkly, thousands of developers working in different languages and frameworks can follow a standard release pattern—one in which speed, risk mitigation, and compliance are embedded. A number of platform engineering teams are making feature management an essential piece of the software delivery pipeline within their platform.

LaunchDarkly is a core part of our software delivery pipeline transformation.

Director of Enterprise Engineering, Fortune 500 financial services firm

Author

Matt DeLaney
Senior Product Marketing Manager
LaunchDarkly

Conclusion

Change management has its place. But when misapplied and overprescribed, it hinders innovation. Research from Gartner, Forrester, DORA, and other firms suggests that, in the context of shipping software, heavy change management undermines business performance.

Nevertheless, organizations will cling to change management and put off continuous delivery until their concerns around operational health, security, and compliance are addressed. Feature management offers the safeguards needed to accelerate software delivery in a low-risk, compliant fashion.

With LaunchDarkly feature flags, you can confidently decentralize software change approvals. You can enable change managers and IT operators to devote more energy to strategic projects. You can pursue continuous delivery without fearing the risks and regulatory implications as much. You can automate change controls within deployment and release workflows. And finally, you can advance platform standardization with feature management as a bedrock capability.

Feature flags aren't a panacea. But for many, their impact on software delivery is nothing short of profound.

Empowering all teams to deliver and control their software.

launchdarkly.com

sales@launchdarkly.com

LaunchDarkly