LaunchDarkly

Feature Flags for Beginners

What they are, how they help, and when to use them

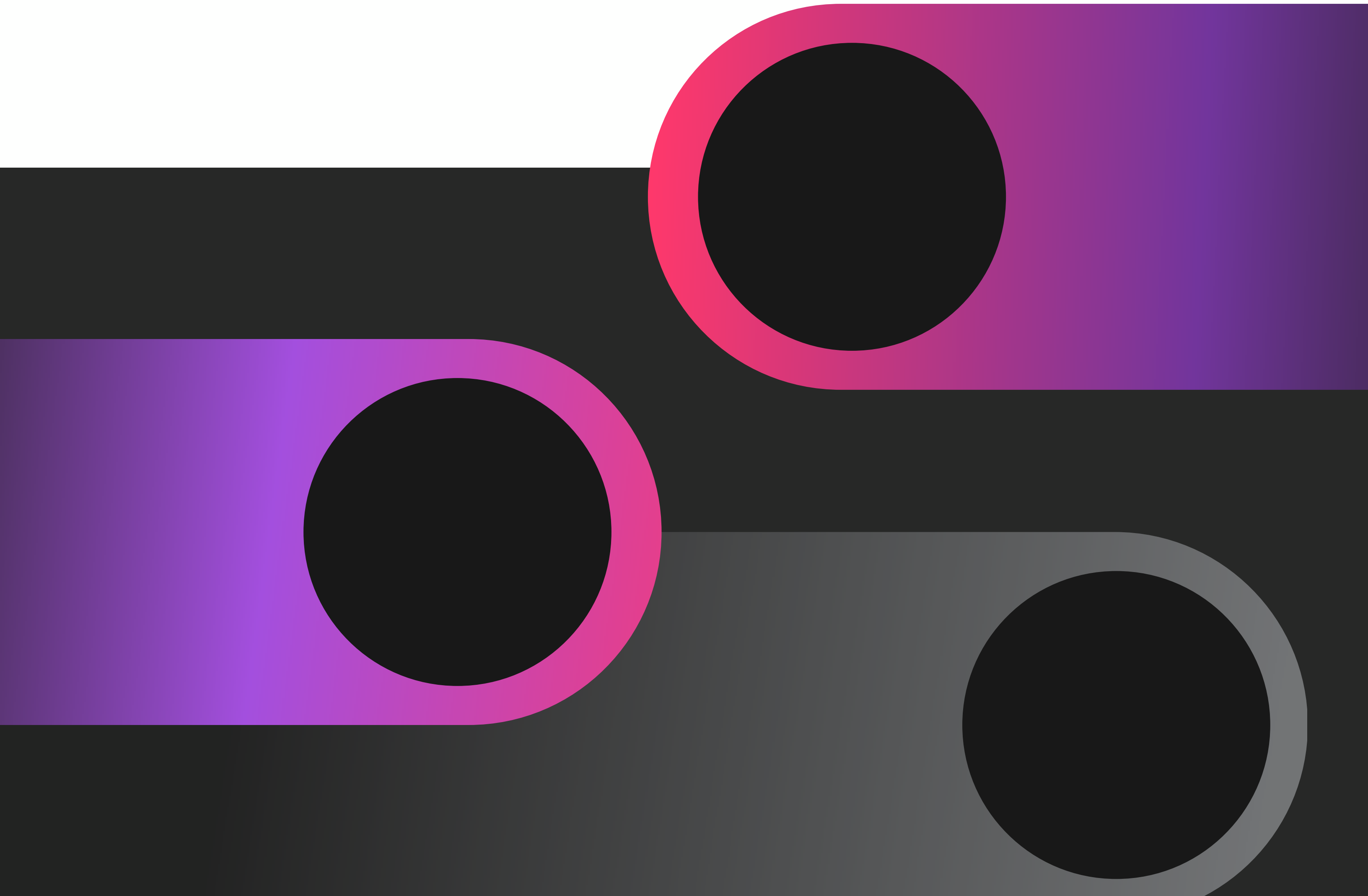


Table of Contents

Introduction

Software development is driven by iterations. Product managers sift through feedback and define business requirements. Developers and designers deploy updates that satisfy users' needs. Data scientists observe user behaviors and collect feedback, kicking off the cycle once more. This is how an idea becomes software that people use.

Of course, when you're involved in the software development process, things never feel so simple. Even the most well-tested code can't account for every real-world situation. More often than not, bugs in production create surprising issues. Monitoring can certainly report on whether a feature has unexpected consequences, but solving these errors requires deployment rollbacks and timing hotfixes.

Even outside of quality assurance, there are many reasons why a deployment might stall. For example, teams could have trouble deciding the best approach for building a new feature. A user experience study can help guide them towards building solutions people want, but these studies can often take too long to set up and assess.

In these cases (and many others), using feature flags is a powerful strategy for quickly surfacing what works—and what doesn't.

In this guide, we'll introduce you to the concept of feature flags, why teams are using them, best practices, and some things not to do with them.

What are feature flags?

A feature flag is a software development process used to enable or disable functionality remotely without deploying code. New features can be deployed without making them visible to users. Feature flags help decouple deployment from release, letting you manage the full lifecycle of a feature.

They eliminate the need to maintain multiple feature branches within your source code. All code changes can be made to the primary branch; only when the feature is ready will it be enabled via the flag.

For example, if you were redesigning your app and wanted to see how people would react to the changes, you could show the new design to only half of your active users. Or, if you built some functionality aimed at enterprise teams, you could target only those users who belong to a large organization.

You may have seen feature flags referred to by other names, such as:

- Feature toggle
- Feature flipper
- Conditional feature
- Feature switch
- Feature controls
- Release toggle

Regardless of the name, their purpose is to use known information to control how code works.

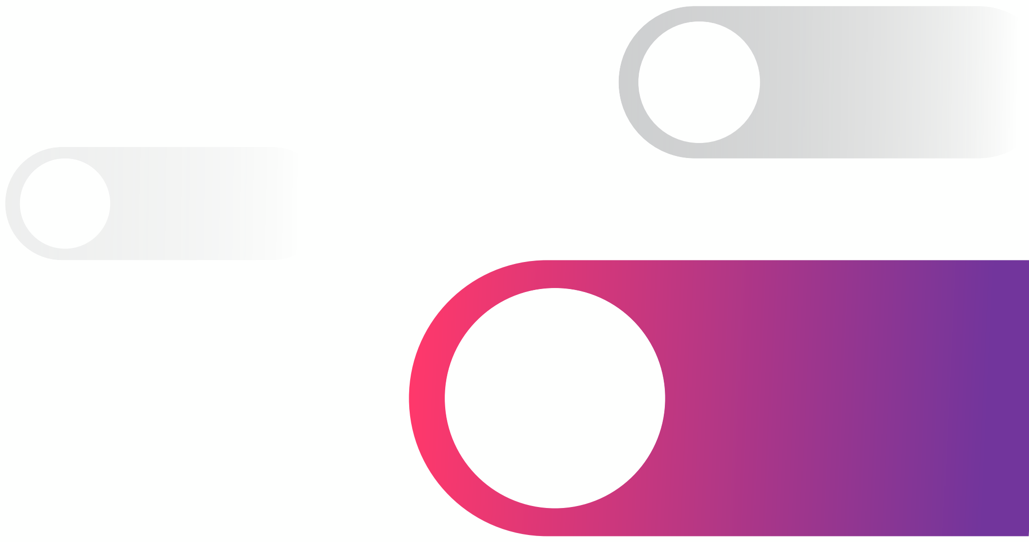
Beta programs and tier access control

You can even run a beta program with feature flags. A user could sign up to get access to a new, rough-around-the-edges feature, and a target feature flag would grant them access. This option takes the burden of maintaining a feature flag infrastructure off of your Ops teams, allowing them to focus on the more critical needs of your application.

Whenever a user navigates to a new page or route in your app, you can check for the value of the feature flag and choose to act on it. Conceptually, this flow is not much different than other authorization logic your app might have.

Consider, for example, a blogging site with support for both free- and premium-tier writers. Free-tier writers would have a limited number of features; meanwhile, premium-tier writers have access to additional features, such as the ability to customize their design. You can control the capabilities accessible only to paying customers with a feature flag.

For a simple case like this, a feature flag backed by a database might make sense, and cover this a bit later in the guide. However, the advantage of feature flags deployed through a feature management platform is that they scale much more easily, helping your organization to stay on top of flag maintenance and technical debt.



Who uses feature flags?

Feature flags aren't just for engineering or development teams. Flags can be used by:

- Sales and Customer Support to provision entitlements.
- Product Management to manage beta programs.
- Marketing to run A/B tests.

Feature management is commonly used at companies of all sizes and across all verticals that are looking to:

- Reduce the number of code rollbacks.
- Gradually roll out new functionality to users.
- Minimize risk by first releasing to small groups of users.
- Test user acceptance.

Companies using continuous integration/continuous delivery (CI/CD) or Progressive Delivery rely on feature flag management to gradually roll out features. **Continuous delivery** gives you the ability to shorten release cycles and get new functionality in the user's hands quickly and safely. **Progressive Delivery** kicks it up a notch and provides fine-grained control over the rollout and ownership of a feature. Progressive Delivery includes user segmentation, traffic management, observability, and automation to release features to small groups of users.

Through these delivery processes, teams develop a cadence for lean releases, where new code is wrapped in feature flags. Features are gradually rolled out to the user base to validate their integrity and minimize the risk to your platform and company.

The advantages of feature flags

For every new feature that's developed, it's common practice to write test cases with expectations for how your app ought to behave. This ensures that future changes to your evolving codebase don't cause disruptions for your users. However, no matter how well-defined your integration and acceptance test coverage might be, once code lands in production, there is always the possibility for unexpected bugs or issues to arise.

Real users in production can reveal more about how a feature works than any test ever could. For example, you may have neglected to optimize your database queries. This is a serious performance problem that only manifests when you have many active users and records. Or, you might expect you have enough CPU, memory, and disk space to run your app efficiently, only to find that your new feature is extremely popular, consuming all your available resources.

Gradually bringing users in through feature flags can help uncover—if not entirely mitigate—these issues. If you notice your database cannot keep up with the activities of your feature-flagged users, you can address the problem immediately, without incurring downtime for everyone. Feature flagging can help you monitor how your infrastructure handles an influx of new activity in scenarios you can't predict.

Ultimately, feature flagging is about being careful, considerate, and controlled. Software can indeed be fickle, but as long as you know that, you can introduce changes slowly, with minimal fear of breaking something.

When to use feature flags

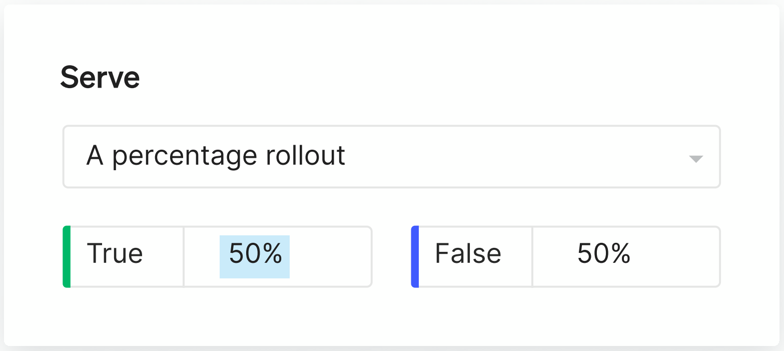
There are four main categories of when to use feature flags.

1 Improve feature rollout (release management)

Using flags as part of your build or feature release process is one of the most common uses. Release management includes early access programs, canary releases, and beta programs—anywhere you give specific users access to features before releasing the feature to everyone.

Starting small and rolling out to larger groups over time helps you:

- Observe the behavior of the systems and services under increasing load.
- Collect user feedback and make changes if you need to.
- Limit the blast radius if something goes wrong.



Serve

A percentage rollout

True	50%
False	50%

2 Operational efficiency

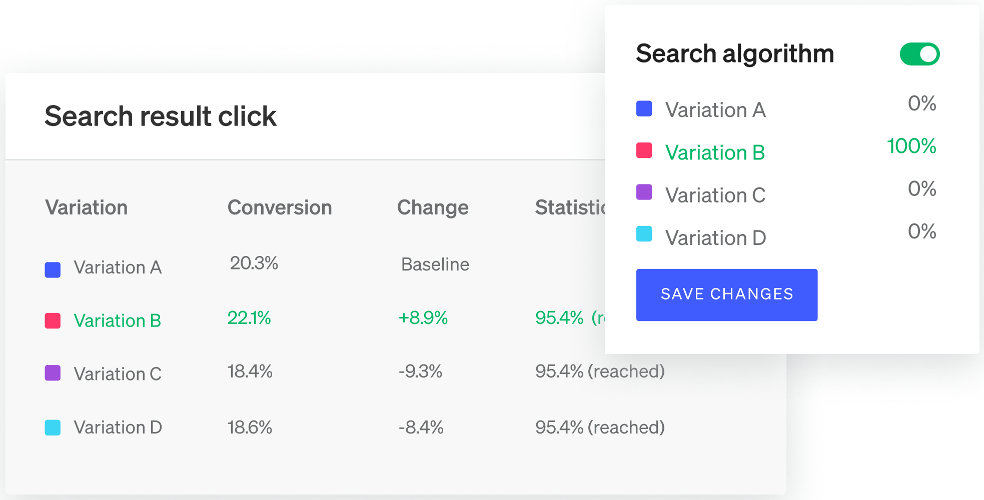
Feature flags can manage and control the behavior of a system by toggling a feature on or off to minimize the impact of incidents.

By using operational feature flags, you can:

- Deploy circuit breakers or kill switches to programmatically or manually disable a feature if it negatively impacts the user experience and triggers alerts.
- Limit API requests to ensure API reliability.
- Switch to a less feature-rich version of a page under heavy load.
- Perform interoperability testing on new microservices or third-party tags in production.
- Change log levels on the fly for debugging purposes.

3 Learn from experimentation

A large part of building and operating software is learning what works and what doesn't. Experiments are one way of learning about your software, infrastructure, users, and systems. Experiments include concepts such as A/B testing, game days, or chaos experiments, all of which allow you to gain new knowledge. When using feature flags, you can test different configurations of a feature to validate or disprove gut feelings. Experiments provide concrete reporting and real-world measurements to ensure that you are launching the best version of a feature for both your users and company metrics.



Search result click

Variation	Conversion	Change	Statisti
🟦 Variation A	20.3%	Baseline	
🟥 Variation B	22.1%	+8.9%	95.4% (r
🟪 Variation C	18.4%	-9.3%	95.4% (reached)
🟦 Variation D	18.6%	-8.4%	95.4% (reached)

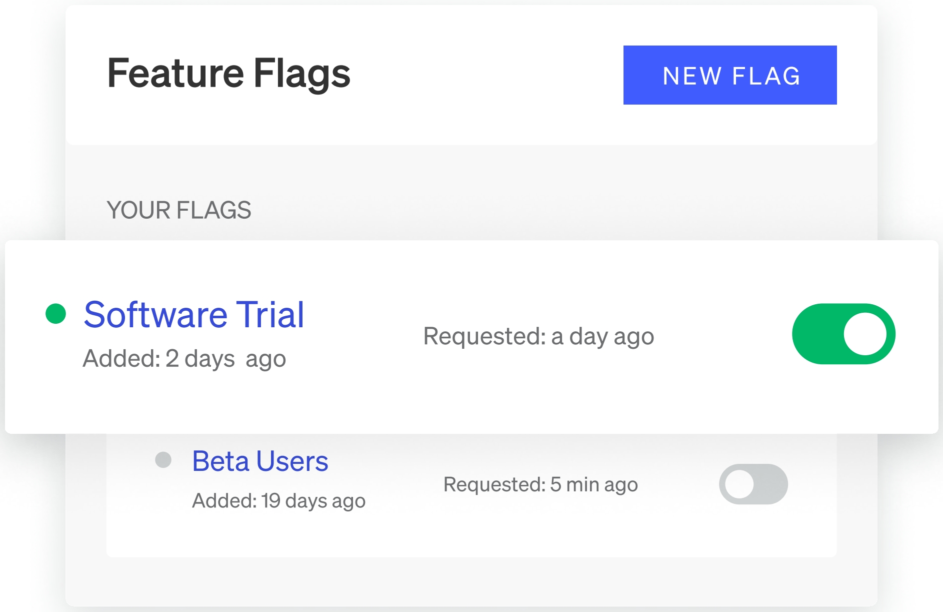
Search algorithm

🟦 Variation A	0%
🟥 Variation B	100%
🟪 Variation C	0%
🟦 Variation D	0%

4 Empower teams with entitlements

In software, entitlement refers to the right to use services, products, or features. Entitlements define the features each user can access. They are sometimes referred to as paywalls—features that are only available to paid subscribers.

Feature flags, in the context of a feature management platform, can empower others in the organization to manage and control entitlements—the right to use services, products, and features. Feature flags can help you streamline the process of managing entitlements, resulting in better customer experiences and greater operational efficiency. Read here for more.



Examples of feature flags in action

Looking for inspiration on where you can use feature flags? Check out these unique examples:

A restaurant chain deployed feature flags to switch between POS vendors if the system crashed or became unavailable, thus minimizing lost revenue and unhappy customers when orders could not be processed and paid for.

At a SaaS company, assigning support tickets was often a manual process. Feature flags were used to determine the capacity of technical support engineers and leads, so each ticket could be assigned to the engineer with the most capacity. Automating ownership based on capacity reduced the amount of time before a ticket was in the hands of a support engineer.

An automotive marketplace, when migrating from on-premises data centers to the cloud, needed to move data in small increments to check for errors. Feature flags helped them selectively route small percentages of traffic based on categories.

How are feature flags deployed?

How you choose to deploy feature flags depends on several factors, including:

- The size of your engineering team
- How often you release code to production
- How many flags you want to deploy
- Whether you want to build or buy

Based on these factors, your options for deploying feature flags start with using a configuration file to store global application settings. If a single feature needs to be “on” or “off” for all users, putting it in a config file is a straightforward way to deploy.

If you have more than a couple of flags or you need more granular controls, you may opt to deploy flags via a database. When you manage flags from a database, features can be targeted to individual users. You can also make modifications to user targeting without having to restart or re-deploy services.

You may opt to deploy using an open source solution. Open source solutions allow for user segmentation, targeting, and controlled feature rollouts, but they are often specific to a language or stack. If your application uses multiple languages, you may need to deploy multiple solutions for complete coverage.

You may also use some sort of homegrown feature flag system that pulls from various elements of the above and addresses needs specific to your company. The challenge with homegrown systems is your team will end up babysitting the tool and performing all its maintenance, and if you go through a period of scale you'll likely run into issues.

A final deployment option is to use an enterprise feature management solution like LaunchDarkly. When you have needs such as role-based access controls (RBAC), audit logs, or use cases including experimentation or entitlements, it’s best to deploy a comprehensive feature management solution.

Learn more about feature flagging

Feature flagging might sound like a complex topic, but it doesn't have to be. You can learn more about how to create your first feature flag or read case studies from companies who have embraced the technique. If you're curious about how to go about implementing feature flags in your app, we've provided some best practices to help guide you on the journey.

LaunchDarkly offers a feature management platform designed for the entire organization. We believe the effective use of feature flags is an organization-wide endeavor, not something limited to a few developers. All teams across the organization can benefit from the use of feature flags. LaunchDarkly grows with you as your feature flagging practice expands from a single flag deployed by a single developer, to team-wide use, to delegating control of flags to other teams to meet their use cases.

To see for yourself, request a demo of the LaunchDarkly feature management platform or start a free trial.

Request a demo

LaunchDarkly

launchdarkly.com