eBook

Feature Flags: Should I Build or Buy?

CloudBees

Table of Contents

Introduction

Building quality software requires finding ways to simplify the development process. Feature flags, or feature toggles, provide a simple way to improve software development. They are simple conditionals used to control code execution.

The concept is intuitive for most developers. If you used a true/false statement to ensure certain parts of your code were turned on or off during runtime, then you were basically using a feature flag. These flags are excellent tools for risk mitigation. Even though the concept is simple, implementing them in a large-scale system can get complicated.

Naturally, the question arises: should you build or buy? There are viable options for both. Before diving into the complexity of choosing a path, let's look at the value these flags provide.

Why are feature flags valuable?

Today software is developed and deployed in increments. The days of massive releases are over. Development managers use steady feature rollout to ensure a calm and safe development process. The flags help in the following areas:

Development Speed

Developing multiple features in parallel is a common software practice. In order to avoid conflict, developers work in their respective feature branches. This process works well until it is time to merge.

Merging branches at different levels of development maturity can turn into a nightmare, and is commonly known as “merge hell” by developers. It takes time and resources to clean or eliminate partially-developed features before shipping. It can slow down the overall development process.

Feature flags decouple features from each other. Instead of using branches, developers can use the toggles to turn off incomplete features from the product. Through simplifying the process, feature flags speed up development.

Deployment Safety	Continuous integration and continuous delivery help development teams deploy faster. Release maturity models dictate the level of automation. However, automation always has an associated risk factor. No automation is fool-proof against catastrophic failure. Feature flags can provide a safety net. The flags can be used to turn off harmful code until a solution is found. It can be even used to regulate experiences based on criteria like user preference or location. It gives a development team more power to control the software experience in the real-world.
Experimentation	In the old model, releases were held sacred. Feature rollout was homogenous for all users. Every user's software experience was the same. Today it's possible to use flags to control the user experience through segmentation. It means experimenting with different business goals is easier. Using A/B testing, you can find out more about your customers. The experimentation can lead to a more satisfactory experience for your users and a more fulfilling creative experience for your developers.
Feature Customization	Some businesses are taking the idea of experimentation to the next level. They want to provide customized versions of their software to different segments of the population. For example, customers who express interest in a particular feature of a software will get it with that feature turned on, while others will never know that feature even exists.

What are the challenges of a feature flag system?

There’s no doubt that your development and production team will benefit from having a feature flag system. You have two options: build or buy. You have to decide what option is best for you depending on your resources, organizational structure, and development process.

Challenges of Building Your Own Feature Flag System

Building your own system gives you full control over all functionalities. It also has challenges:

Lack of visibility and governance	Maintaining a system that meets the need of worldwide teams require access control and auditing capabilities. Internal systems that don’t have these capabilities have a hard time knowing what features are flagged, who has the ability to flag any feature, and who is and isn’t following established protocols for flagging.

Enterprise scalability	Feature flagging usually starts small with one development team, but then expands to more stakeholders and global teams, such as product managers (for the purpose of experimentation), and they would have totally new and different requirements. Most homegrown systems will struggle to support hundreds or thousands of developers flagging at once.
Continuous Infrastructure and Maintenance Cost	You build it, you own it. It means developers, QA and management have to invest time and money to maintain and upgrade the system. The resources diverted to create, maintain and upgrade can take away the focus from your core business.
Lack of Advanced functionalities	Internal flag systems are built for immediate needs. They are generally minimum viable products. They lack advanced features (such as advance release segmentation or gradual release) because of the associated cost. They would usually lack a good UI dashboard that non-technical users could operate, and would then require constant engineering support.
Uncertain and unpredictable requirements	Pivoting and changing requirements are common in the software industry. Building a flag system can be difficult in such an environment. Today's essential custom flag system can become useless tomorrow. You'll be stuck with a dinosaur in the room.

Questions to Ask Yourself

Given the potential challenges, it is important to ask yourself the following questions before deciding to build a system:

1	Do I have the resources to spare from my core business to create a flag system?
2	How much am I willing to spend on the maintenance of the system?
3	What hardware and software investment do I need to create the system?
4	Will I be able to scale my system for other development teams and PMs?
5	Will the flag system be compatible with my current production pipeline?
6	How will we automate the cleanup of flags to avoid massive tech debt?
7	What happens if our flag architect leaves the company?
8	What is the long-term plan for the system?

Considerations for Building a Feature Flag System

After answering the above questions, you will have a better idea of where you stand in terms of building the system. If you decide to build, you need to think about the following:

Dashboard

Your business needs to be able to visualize the feature flagging process and be friendly to non-technical employees. It is not enough to document flags and expect the non-developers to figure it out on their own.

Permissions and auditability

In order to control the flags, you need to set up a system that can handle different levels of permission for different users and ensure there is a trail for auditability. Your admins and your general users should be able to function side by side.

Analytics

Analytics can help determine how the flag system affects performance.

Reliability

Your feature flag system needs to be robust enough to support a production environment.

Considerations for Buying a Feature Flag System

As you can see, building a feature flag system requires a significant investment in time and human resources. Due to the complexity of building a system, you might consider purchasing a feature flagging management platform.

Before you invest in a system, you need to understand if it is going to be a good fit for your organization. You can run pilot projects to get a better idea of its viability. Here are a few other things to look for:

Ease-of-use

Feature flagging systems come with an overhead, you want to make sure the system you choose helps your developers streamline the processes associated with adding/ managing and removing of flags.

Reliability

Ensure the system will not become a bottleneck through introducing its own problems and bugs into the development cycles.

Scalability

Find out if performance degrades under high-volume conditions.

Well-Documented Support

Good documentation can make the integration process easier.

Summary

Choosing to build or buy your feature flag system depends on your business's unique needs and resources, but either path benefits from a clear understanding of its value. A robust feature management approach can accelerate innovation, reduce risk, and enable faster feedback loops.

CloudBees Feature Management, built on the open and extensible CloudBees Unify solution, offers a secure and scalable way for developers to confidently control feature rollouts in production. As part of a broader DevOps solution, it enables progressive delivery within a unified pipeline, integrates seamlessly with your existing toolchain, and provides governance across the entire software delivery lifecycle, eliminating silos and helping you ship smarter, not just faster.

Learn more cloudbees.com/capabilities/feature-management

CloudBees

CloudBees, Inc.
cloudbees.com
info@cloudbees.com