Feature flags

How to progressively expose your features with feature flags.

BY IAN BUCHANAN

Summary: *Feature flagging (also commonly known as feature toggling) is a software engineering technique that turns select functionality on and off during runtime, without deploying new code. This enables teams to make changes without pushing additional code and allows for more controlled experimentation over the lifecycle of features.*

If you plan to continuously integrate features into your application during development, you may want to consider feature flags. After all, you may want to toggle features and hide, disable, or enable them. You may also want to reveal different variations of features to users in order to discover which one is better. Feature flags, also known as “toggles”, “bits”, “flippers”, or “switches” allow you to do this and more.

What are feature flags?

Feature flags (also commonly known as feature toggles) is a software engineering technique that turns select functionality on and off during runtime, without deploying new code. This enables teams to make changes without pushing additional code and allows for more controlled experimentation over the lifecycle of features. Because of this, feature flags enable many novel workflows that are incredibly useful to an agile management style and CI/CD environments.

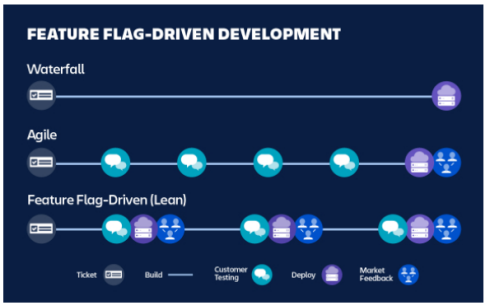
During development, software engineers wrap desired code paths in a feature flag. The following is an example of a basic feature flag written in Javascript:

```
if(featureFlags['new-cool-feature'] == true){
    renderNewCoolFeature();
}
```

This code demonstrates a simple statement that checks if a “new-cool-feature” is enabled. Even with advanced frameworks and tools that help manage flag data or injection and removal of the new logic path, feature flags are essentially just "if statements". Therefore, this binary switch is what all the synonyms have in common.

When this code is ready to ship to production it will be deployed as usual. However, it will be inactive on production until the feature flag is explicitly activated. Flags can be assigned to a subset group of users allowing highly targeted behavior.

Feature flag benefits



At a foundational level, feature flags enable code to be committed and deployed to production in a dormant state and then activated later. This gives teams more control over the user experience of the end product. Development teams can choose when and to which users new code is delivered.

Validate feature functionality

Developers can leverage feature flags to perform “soft

Developers can leverage feature flags to perform "soft rollouts" of new product features. New features can be built with immediate integration of feature toggles as part of the expected release. The feature flag can be set to "off" by default so that once the code is deployed, it remains dormant during production and the new feature will be disabled until the feature toggle is explicitly activated. Teams then choose when to turn on the feature flag, which activates the code, allowing teams to perform QA and verify that it behaves as expected. If the team discovers an issue during this process, they can immediately turn off the feature flag to disable the new code and minimize user exposure to the issue.

Minimize risk
Building on the idea of soft rollouts discussed above, industrious teams can leverage feature flags in conjunction with system monitoring and metrics as a response to any observable intermittent issues. For example, if an application experiences a spike in traffic and the monitoring system reports an uptick in issues, the team may use feature flags to disable poorly performing features.

Modify system behavior without disruptive changes
Feature flags can be used to help minimize complicated code integration and deployment scenarios. Complicated new features or sensitive refactor work can be challenging to integrate into the main production branch of a repository. This is further complicated if multiple developers work on overlapping parts of the codebase.

Feature flags can be used to isolate new changes while known, stable code remains in place. This helps developers avoid long-running feature branches by committing frequently to the main branch of a repository behind the feature toggle. When the new code is ready there is no need for a disruptive collaborative merge and deploy scenario; the team can toggle the feature flag to enable the new system.

Feature flag use cases

The novel utility of feature flags enables a variety of creative uses cases for industrious teams. The following examples highlight some popular applications of feature flags in an agile environment.

Product testing
Feature flags can be used to gradually release new product features. It can be unclear upfront if a proposed new feature will be adopted by users and is worth the return on investment. Teams can release a new product feature or a partial idea of the feature in a feature flag and deploy it to a subset of users to gather feedback. The subset of users may be vocal power users who are happy to beta test and review. If the new product idea proves to be a hit, the development team can then roll out the feature flag to a larger user base. If instead the new product idea turns out to be a flop, the development team can easily disable the feature flag and later remove it from the codebase.

Conducting experiments
Experiments or A/B testing are a primary feature flag example. In their simplest form, feature flags act as a toggle of “on” and “off” state for a feature. Advanced feature flags utilize multiple flags at once to activate different experiences for subsets of users. For example, imagine you divide your user base into thirds. Each third receives its unique flag and user experience. You can then measure the performance of these three flags against each other to determine the final committed version.

Migrations
There are times when an application needs a data migration that requires dependent application code changes. These scenarios are sensitive, multi-phase deployment tasks. A database field may be modified, removed, or added in an application database. If the application code is not prepared for this database change, it causes failures and errors. If this happens, it requires a coordinated deploy between the database changes and the application code.

Feature flags help to ease the complexity of this scenario by allowing teams to prepare in advance application changes in a feature flag. Once the team makes the

changes in a feature flag. Once the team makes the changes to the database, they can immediately switch toggle the feature flag to match the application code. This removes the risk and delay of waiting to deploy the new application code and possibly having the deployment fail and desync the application from the database.

Canary launches
A canary in this context refers to an old, morbid practice where coal miners brought canary birds into the coal mine to detect carbon monoxide. The birds have higher metabolism rates and rapid breathing rates, which led them to succumb to carbon monoxide before the miners.

Canary launches in software development occur when a new feature or code change is deployed to a small subset of users to monitor its behavior before releasing it to the full set of users. If the new feature shows any indication of errors or failure, it is automatically rolled back. Feature flags are essential to this process since they restrict the audience pool and can toggle off features easily.

System outage
A feature flag can also be used as a system outage tool. A web application may utilize a feature flag to “switch off” the entire website for maintenance or downtime. The feature flag can be instrumented throughout the codebase to push sensitive transactions and display outage content to the end users. This can be incredibly helpful when doing sensitive deployments or if an unexpected issue is found and needs to be urgently resolved. It gives teams the confidence and capability to take a controlled outage if deemed necessary.

Continuous deployment
Feature flags can be used as an integral component to build a truly continuous deployment system. Continuous deployment is an automated pipeline that takes new code from developers and automatically deploys it to production end users. Continuous deployment depends on layers of automated tests that verify new code behaves as expected against a matching specification as it moves through the pipeline.

Feature flags make it safer to deploy continuously by separating code changes from revealing features to users. New code can automatically merge and deploy to production and then wait behind a feature flag. The continuous deployment system can monitor user behavior and traffic, then automatically activate the feature flag. Inversely, the continuous deployment system can monitor the new feature flag code to see if it behaves as expected, and roll it back if not.

Feature branches vs. feature flags

How to implement feature flags

There are many paths to implement feature flags with varying logistical considerations and return on investment. The path to take depends on your team's needs and organizational goals.

Feature flagging has some infrastructure dependencies that need to be addressed to function properly. As teams scale their use of feature flags and the switching on/off of flags becomes a business decision, it becomes critical to have an authoritative data store and a management mechanism for the flags. Many third-party feature flag services provide this data store dependency.

Third-party hosted feature flag services are often the best solution. They handle the heavy logistics, and offer easy-to-integrate libraries that expedite the installation process. This allows teams to focus on core business duties instead of infrastructure management. However, if your team has third-party security concerns, it may be in your best interest to implement your security flag backend.

Separately, engineers need to instrument new code logic that retrieves the flag state from the service to activate the flagged content. This requires code merges and deployments of the flag code before it’s activated. Since many feature flags are temporary, don't forget to remove the feature flags that are no longer necessary.

In conclusion...

Feature flags are a powerful addition to an agile development arsenal. There are many creative ways to utilize feature flags. Feature flags are complementary to continuous deployment and Git version control. Overall, feature flags give teams more control of their codebase, their deployment, and the end-user experience.

IAN BUCHANAN

While Ian has broad and deep experience with both Java and .NET, he's best known as a champion of agile methods in large enterprises. He's currently focused on the emerging DevOps culture and the tools for enabling better continuous integration, continuous delivery, and data analysis. During his career, he's successfully managed enterprise software development tools in all phases of their lifecycle. He has driven organization-wide process improvement with results of greater productivity, higher quality, and improved customer satisfaction. He has built multi-national teams that value self-direction and self-organization. When not speaking or coding, you can find Ian indulging his passions in parsers, meta-programming, and domain-specific languages. Follow Ian at @devpartisan.

Bitbucket CI/CD tutorials

Continuous Delivery articles

ARTICLE

Feature Branching Workflows for Continuous Delivery

Using feature branching workflows in your continuous delivery pipeline keeps your most important branches in a clean and releasable state and allows develo

Read this article →

ARTICLE

Super-Powered Continuous Delivery with Git

Now that Git has solved the pain of merging, learn more about branching workflows and best ways to do continuous delivery with Git and branching techniques

Read this article →

CI/CD Topics

Continuous Delivery
Continuous Integration
Continuous Deployment
Pipelines
Software Testing
Microservices
Tutorials

Sign up for more CI/CD articles and tutorials.

Email

email@example.com

Subscribe

Up Next
Platform as a Service →