Decoupling Deployment and Release- Feature Toggles

Abhishek Tiwari

Citation: *A. Tiwari*, "Decoupling Deployment and Release- Feature Toggles", Abhishek Tiwari, 2013. doi:10.59350/6ayra-sc056

Published on: October 18, 2013

There are key differences between deployment and release. Deployment means putting the latest stable version of code on your web application servers. Latest version of code often includes a range of features or improvements. Release means making all or selected features available to the public or end-users. In addition, one main difference between deployment and release is the ability to roll back.

Think in this way, when a retailer moves new range of products from warehouse to a store - that is a deployment. When these new products go on store shelves so that people can buy it in store - that is a release. Now retailer may delivery 100 new products from warehouse to store, but store manager may opt to put only 10 products on the store shelves. That is one way of doing selective release. Another way to do a selective release will be to offer these 10 products to VIP customers only.

Decoupling

In a new world most of web-scale companies have decoupled the deployment from release process. Decoupling the deployment from the release helps to achieve the zero-downtime release. With Zero-downtime release, switching user from one release to another happens instantaneously. It also helps to instantaneously rollback to previous release when required. This decoupling also enables dev and ops teams to deploy new versions of application continuously, completely independent of the decision as to which features are available to which users.

Phased rollout

Now a days when Apple starts rolling out a new version of iOS, although latest release is deployed on all delivery servers but not everyone can download latest release. To get the latest iOS release, you have to be part of a queue. By doing selective release Apple achieves two things - a better capacity management and identifying and fixing the bugs before it opens door for wider audience.

Similarly when Facebook or Google+ start rolling new features they release it in a phased manner - first releasing to a selective audience and progressively to everyone else. It also enables Facebook and Google+ to test things on early users and react to responses.

Dark Launching

Term dark launching was coined by Facebook engineering team to simulate the new features in production environment well before release.

> The secret for going from zero to seventy million users overnight is to avoid doing it all in one fell swoop. We chose to simulate the impact of many real users hitting many machines by means of a “dark launch” period in which Facebook pages would make connections to the chat servers, query for presence information and simulate message sends without a single UI element drawn on the page. With the “dark launch” bugs fixed, we hope that you enjoy Facebook Chat now that the UI lights have been turned on.

Using a dark launching approach you can soak test new features and fix any bug detected during dark launch period. Dark launching can also be used to prepare application for release by

- pre-caching clients side assets and libraries in user browser
- warming up application side caching
- performing database migration, expansion only

Feature Toggles

Feature toggles are used for dark Launching or phased rollout. Once latest version of application deployed we can toggle and expose a set of features to selected or all end-users. Feature toggles can be also used for,

- for avoiding branching and merging
- experimenting such as A/B tests
- putting unfinished code in production
- reducing risk associated with large change
- turning a resources heavy feature OFF in high load conditions

According to Ross Harmes @ Flickr

> Feature flags and flippers mean we don’t have to do merges, and that all code (no matter how far it is from being released) is integrated as soon as it is committed. Deploys become smaller and more frequent; this leads to bugs that are easier to fix, since we can catch them earlier and the amount of changed code is minimized.

Martin Fowler’s explanation of feature toggle,

> The basic idea is to have a configuration file that defines a bunch of toggles for various features you have pending. The running application then uses these toggles in order to decide whether or not to show the new feature.

Normally features toggles can have boolean states - ON or OFF. But depending on your requirements a feature toggle can be more than boolean states - lists, arrays, strings, conditions, logical statements, integers or any other value. State of toggles is either set statically using application configuration based toggles or dynamically using rules based toggles.

Once a list of toggles are available in application context or namespace, you can use them your application code using simple `if` conditions.

```
if (MyFeatures.HOT_NEW_FEATURE.isActive()) {
  // do cool new stuff here
}
```

```
<h:panelGroup rendered="#{features['HOT_NEW_FEATURE']}">
  <!-- Some part of the page required for HOT_NEW_FEATURE -->
</h:panelGroup>
```

For frameworks like Rails or Django, you can define feature flipping on all levels - Model, View, Tempaltes and Controller.

Application congifuration based Toggles

Using application configuration file (XML, YAML or JSON), you can switch ON or OFF selected features.

Rule based Toggles

Rule based toggles can be applied to selected records using database queries - filter using a select condition and update toggle state to ON.

> select all users where friends.count > 1000 and set TIMELINE_FEATURE = ON

Another way to apply rule based toggles is to use performance parameters like number of users, average load time etc.

> set MINICART_FEATURE = OFF when live.users > 5,000

> set MINICART_FEATURE = OFF when loadTime > 5s

Performance based toggles can help you to degrade your service under high load conditions gracefully. Similar technique was used when Facebook launched usernames feature.

Rule based toggles are normally created using a admin interface. Facebook uses a similar rule based toggle admin called as Gatekeeper. Gatekeeper works with Facebook's feature toggles to control who can see which features at runtime.

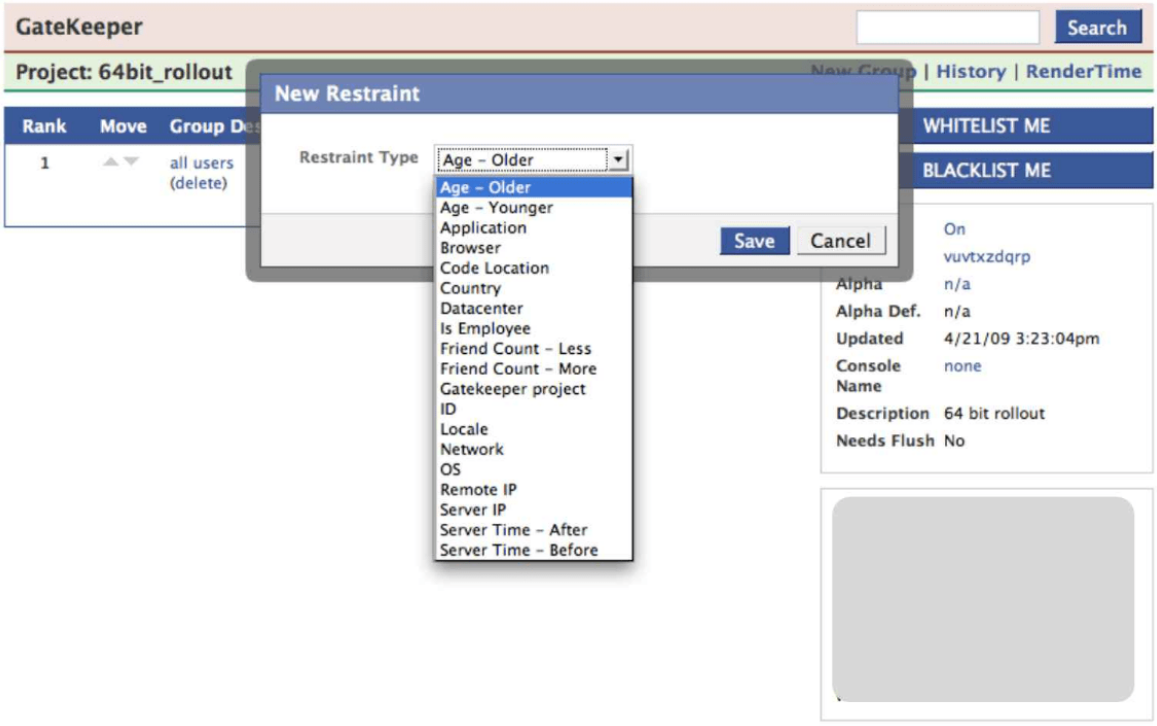

Figure 1: Gatekeeper admin interface to define rule based feature toggles

Feature groups

Once your start using feature toggles regularly, it is good idea to group the features. For instance, you can group all features with positive impact on performance. Moreover, feature groups are very handy when one feature depends on another.

Process, Design and Life Cycle

Feature toggles require a robust engineering process, solid technical design and a mature toggle life-cycle management. Without these 3 key consideration, use of feature toggles can be counter-productive. Remember main purpose of toggles is to perform release with minimum risk, once release is complete toggles need to be removed.

In terms of process, use of feature toggles requires a streamlined and channeled process. First of all there should be clear guidelines, when to use toggles or not. Then there should be a change management control around turning toggles ON or OFF.

Basically you need a CRUD life-cycle for feature toggles, otherwise they will become technical debt. So,

- Development team `create` toggles when required
- Application `read` or expose toggled feature when they are switched ON
- Development team `update` toggled features when they are buggy
- A semi-automated process to `destroy` or remove toggles when job is done

Destroy or remove toggles when a feature,

- is released and no issues are pending or all raised issues are fixed and feature is stable
- was not released due to various issues and fixing issues not possible or may not be high priority

Support for feature toggles

Feature toggles are supported by most of mainstream programming languages and web-frameworks. Most these libraries or packages also offer toggle admin interface to define feature toggles.

Feature toggles in Ruby

- Flip
- Rollout
- Degrade

Feature toggles in PHP

- FeatureToggle (Symfony)

Feature toggles in Python

- Gargoyle (Django)
- Django Waffle (Django)
- Nexus admin

Feature toggles in JavaScript:

- FeatureFlipperJS (Node)

Feature toggles in .net:

- FeatureToggle

Feature toggles in Groovy

- GrailsFeatureToggle

Feature toggles in Java

- Togglz