

# Fantastic Feature Flags 🚩

Paul Heasley | Engineering Guild 28 Sep 2022



A mechanism for easily enabling or disabling certain system behaviour

What is a feature flag, feature toggle, feature switch

# Simplest feature flag is just an if statement

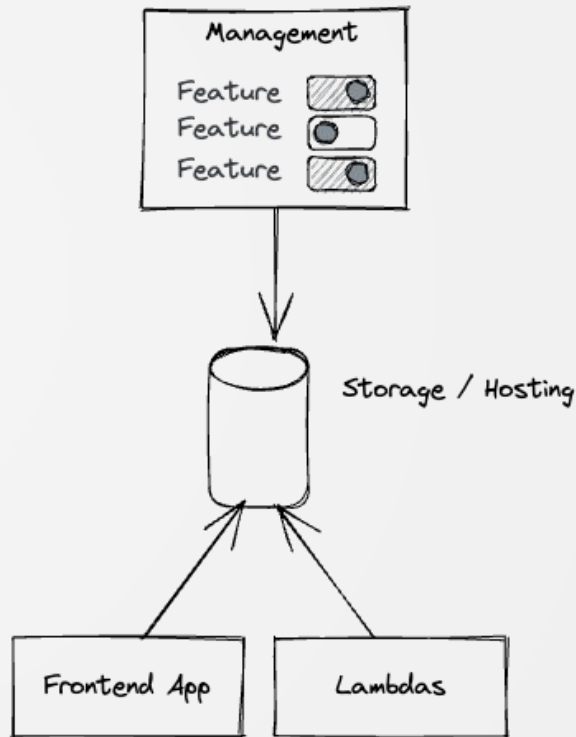
But this requires a developer to change and deploy it.

```
let auth
const useCloudentity = true
if (useCloudentity) {
  auth = cloudentityAuth()
} else {
  auth = cognitoAuth()
}
```

# Ideally we want run-time configurability

Allows us to turn things on and off without code changes.

Allows non-engineers to flip the switch.



We might also want complex behaviour

```
{  
  "name": "myFeature",  
  "description": "A feature that's only enabled for some retailers or users",  
  "enabled": true,  
  "value": {  
    "retailers": [ "retailer1", "retailer2" ],  
    "users": [ "user1", "user2" ],  
    "expires": "2018-01-01T00:00:00Z"  
  }  
}
```

There are many  
hosted services

But they can be expensive 💰 and  
add another possible bottleneck




LaunchDarkly



**split**

# So we built our own



paul@eql.xyz

- Draws
- Live Draws
- Consumers
- Features**

## Features

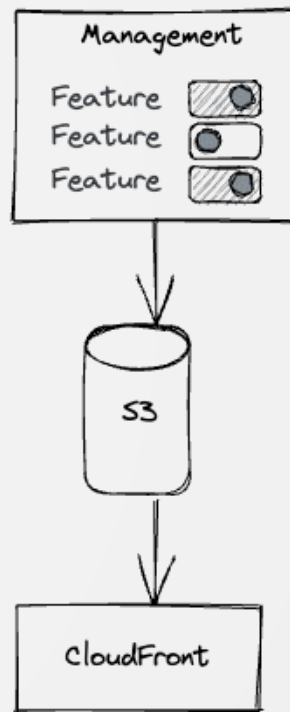
+ Add Feature

<b>bannerMessage</b> A message that can shown at the top of any draw page in case of a system outage.	<input type="checkbox"/>
<b>datadogPremiumSampleRate</b> Datadog Realtime User Monitoring premium (session recording) sample rate @ \$2.60 per 1K.	<input checked="" type="checkbox"/>
<b>datadogRumSampleRate</b> Datadog Realtime User Monitoring sample rate @ \$1.80 per 1K.	<input checked="" type="checkbox"/>
<b>useCloudentity</b> Use Cloudentity as our authentication provider. ** DO NOT TURN THIS ON OR YOU'LL BREAK PROD**	<input type="checkbox"/>

Read more in the Feature Flags Design Doc

# It's just an S3 JSON file with CloudFront

```
{
  "features": [
    {
      "_createdAt": "2022-09-06T02:00:57.278Z",
      "_updatedAt": "2022-09-06T02:03:23.906Z",
      "description": "Datadog Realtime User Monitoring sample rate @ $1.80 per 1K.",
      "enabled": true,
      "name": "datadogRumSampleRate",
      "type": "number",
      "value": 10
    }
  ]
}
```

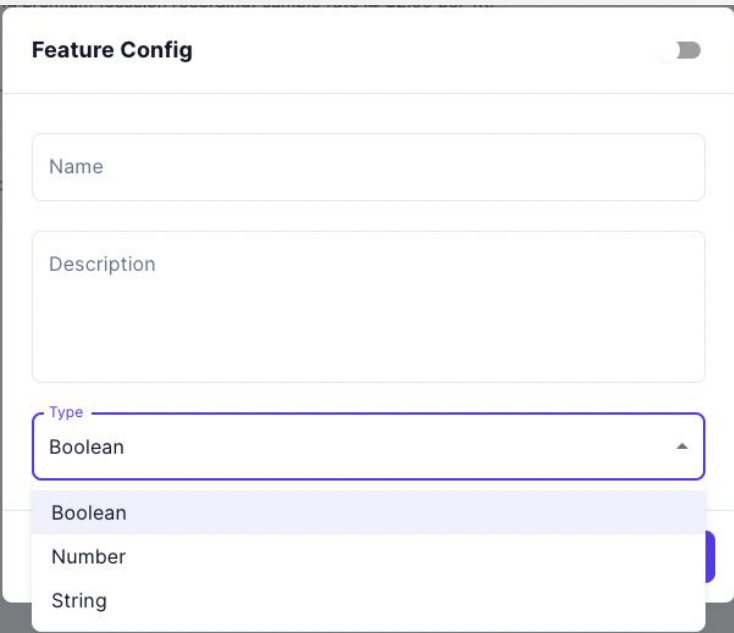


<https://features.eql.xyz/features.json>



# But it's very flexible

Supporting boolean, number or string types



A screenshot of a 'Feature Config' form. It has a title bar with a toggle switch. The form contains three input fields: 'Name', 'Description', and 'Type'. The 'Type' field is a dropdown menu with 'Boolean' selected. A dropdown menu is open below the 'Type' field, showing the options 'Boolean', 'Number', and 'String'. The 'Boolean' option is highlighted.

**Feature Config**

Name

Description

Type

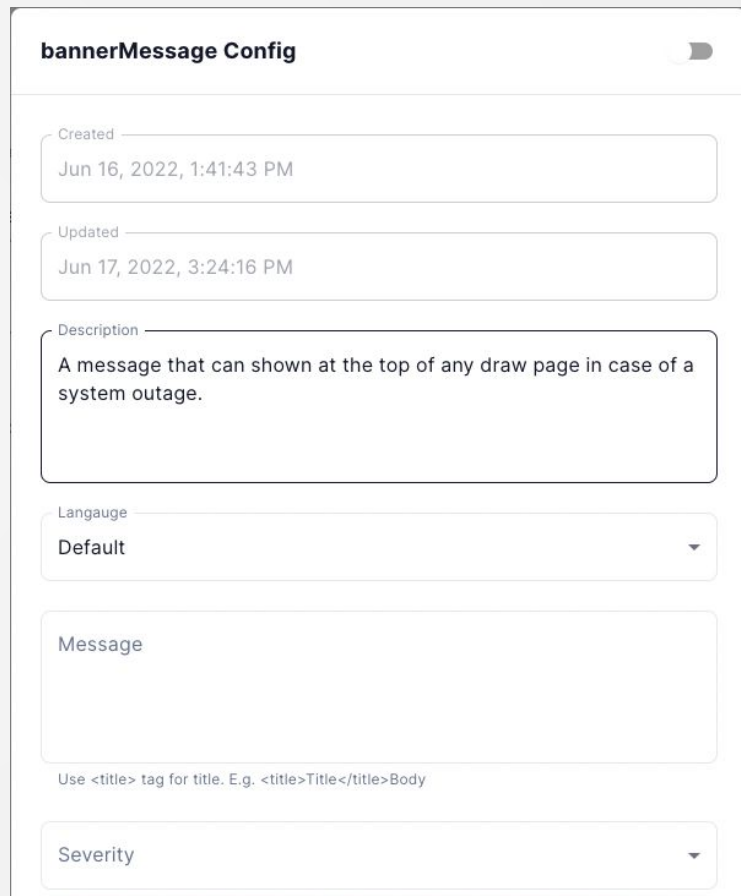
Boolean

Boolean

Number

String

Or custom JSON values



A screenshot of a 'bannerMessage Config' form. It has a title bar with a toggle switch. The form contains several fields: 'Created' (Jun 16, 2022, 1:41:43 PM), 'Updated' (Jun 17, 2022, 3:24:16 PM), 'Description' (A message that can shown at the top of any draw page in case of a system outage.), 'Language' (Default), 'Message' (Use <title> tag for title. E.g. <title>Title</title>Body), and 'Severity'.

**bannerMessage Config**

Created

Jun 16, 2022, 1:41:43 PM

Updated

Jun 17, 2022, 3:24:16 PM

Description

A message that can shown at the top of any draw page in case of a system outage.

Language

Default

Message

Use <title> tag for title. E.g. <title>Title</title>Body

Severity

# There's even a React SDK for consuming it

```
import { useFeatures } from "src/hooks/use-features"

const { features, getFeature, loading } = useFeatures()
// if loading is true, features are still loading, provide some default behaviour

// Access a simple feature value if enabled
const useCloudentity = getFeature("useCloudentity")
if (useCloudentity) {
  // Login with Cloudentity
}
// Or loop through all features and get the whole feature object
const useMailgun = features.find(f => f.name === "useMailgun")
if (useMailgun.enabled === true) {
  // Send emails with Mailgun
}
```

# It's waiting for a backend SDK

As it's just an S3 file, it's easy to read from it within a lambda or even trigger events when the S3 file changes.

Ideally we want to build a caching mechanism if a lambda is reading it frequently.

**Lambda extensions** may be a good solution to pre-load feature flags, similar to [this Secrets Manager solution](#).

# Currently your dev feature file is empty

To enable a feature you need to create a new feature with the correct name and type, e.g. `useCloudentity: boolean`.

A better approach would be a way to sync production features as defaults. We still need to work out a solution to this.

# Why feature flags?

Here's 6 use cases



# Use case 1: Continuous integration & delivery

Feature flags restrict access to certain features or code that is not ready for general availability.

They enable us to **deploy** even when we're not ready to **release**.

## Continuous integration

Code should not live outside of the **main branch** for long.

## Continuous delivery

Code should not live outside of the **production** for long.

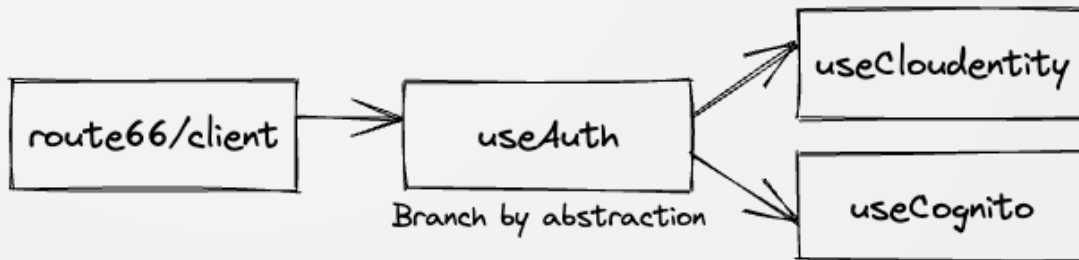
# Long running feature development

## Don't

- Create a complete copy of the service (it will get out of sync)
- Create a long lived feature branch

## Do

- Create a feature flag and commit incrementally and often
- Branch by abstraction



## Use case 2: Gradual release

- Testing in production
- Canary testing
- Dark launching



# Use case 3: Timed release

E.g.

- Before a high heat draw
- Coordinating rebrand / marketing release
- Comply with new regulations (e.g. new tax rates)

## Use case 4: A/B testing

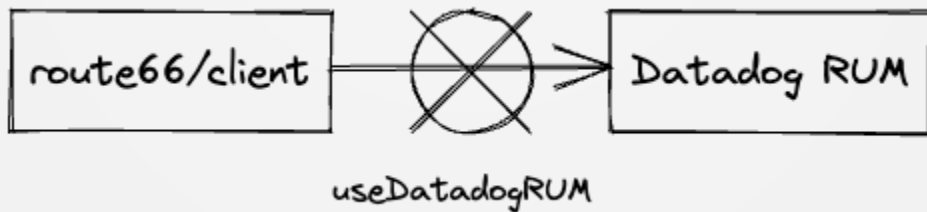
A/B testing is the practice of **serving 2 different variations** of a system (e.g. UI change) to a random distribution of the population and comparing the performance.

Services like Split.io excel at this feature, and it usually accounts for 80% of the cost but it used 2% of the time\*.

\* Numbers derived from Paul's very objective gut feel

## Use case 5: Circuit breakers

Circuit breaker feature flags allow us to **dial down, or turn off** functionality (e.g. Datadog RUM sampling).



## Use case 6: Business features

Using feature flags we can support multiple customers / tenants with varying configurations in a single code base.

# Best practices

Feature flags come at a maintenance cost.

They should be used sparingly and cleaned up as soon as possible.

They should include a description and expiry date when they can be removed.

[featureflags.io](https://featureflags.io) (by LaunchDarkly) is a great resource.

