Engineer

機能フラグ

🏳 再ビルドせずにコードの有効・無効を切り替えるスイッチ

どのようなものか

機能フラグは、アプリを再ビルドせずにチームが変えられる設定を読む、コード内の判断箇所です。その背後には、すでに本番へ配備されていても、設定で有効にするまでは動かない、新しい処理経路があります。全員に有効にすることも、選んだ利用者だけに有効にすることもできます。Pete Hodgsonは、Martin Fowlerのサイトで、2016年1月から連載形式で詳しく説明し、2017年10月に改訂しました。Hodgsonはfeature togglesと呼び、feature flags、feature bits、feature flippersも、同じものの別名として挙げます。

Hodgsonは、フラグをいくつかの部分に分けます。トグルポイントは、新しい経路が有効かを尋ねる、コード内の箇所です。トグルルーターが、トグル設定から答えを決めます。誰が尋ねるかで答えが変わる場合は、トグルコンテキストも使います。通常は現在の利用者で、利用者を表す特別なCookieやHTTPヘッダーを使うこともあります。Hodgsonは、この二つを密結合させないよう警告します。トグルポイントのコードは、この注文確認に新しいキャンセル用リンクを含めるか、といった、対象を絞った質問をするべきです。答えを決める規則は、ルーターの背後の一か所へ置きます。すべての呼び出し箇所に触れずに、規則を変えられるためです。

フラグは、存続する期間と答えが変わる頻度によって異なり、Hodgsonは四種類に分けます。リリーストグルは、未完成で未テストのコードを、休眠状態のまま本番へ配備できるようにします。「通常は、一、二週間を大きく超えて残すべきではない」と述べます。実験トグルは、A/Bテストのために利用者を分け、同じ利用者を毎回同じグループへ入れます。運用トグルは、本番でのシステムの振る舞いを、運用担当者が変えられるようにします。多くは短命ですが、一部の長く使う停止スイッチは、異常な高負荷のときに、なくても動く機能を縮小します。生成にかかる処理の負担が大きい推薦パネルなどです。Hodgsonは、手動で管理するサーキットブレーカーに例えます。権限トグルは、有料顧客向けのプレミアム機能や、自社従業員向けのアルファ版機能など、各利用者に見せる機能を決め、何年も残る場合があります。

機能フラグとカナリアリリースは、よく併用されますが、別のものです。カナリアは、サービスの二つのバージョンを並行して動かし、一部のトラフィックを新しい方へ送ります。フラグは、両方の経路を含む一つのバージョンを配備し、リクエストごとに選びます。バージョン管理の長期ブランチも、新しい仕事を分けておきますが、本番へ届くのは、ブランチをマージしたとき、一度にすべてです。

概念を判断に生かす

「機能フラグとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 3社の間で受け渡す代わりに、1つのチームがアイデアのスケッチから、お客様がログインして使える製品まで担当します。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 GIS地図アプリの開発を主導し売上を10倍に — 10x 地図製品の開発を主導して実現した売上成長

GIS地図アプリの開発を主導し売上を10倍に お問い合わせ →

なぜ重要なのか

フラグを使う主な理由は、配備と公開を分けることです。Hodgsonは、リリーストグルを、継続的デリバリーの原則を実践する最も一般的な方法と説明します。機能の公開と、そのコードの配備は、別の段階です。一つのメインブランチから毎日、本番へコードを出しながら、一か月かけて作る機能は、準備できるまで無効にしておけます。これが、トランクベース開発を可能にします。

フラグは、利用者によって答えを変えられるため、公開範囲を段階的に広げられます。Hodgsonは、まず社内の利用者へ有効にする方法を、Champagne Brunchと呼びます。会社が先に、自分たちのシャンパンを飲むためです。次は、カナリアの対象グループです。例では、利用者IDの剰余で無作為に1%を選び、同じ人がグループに残るようにし、ほかの99%は従来の振る舞いを保ちます。そのグループから問題の報告があれば、フラグを無効にすればよく、配備をロールバックする必要はありません。

代償もあります。フラグごとに、コードの分岐と、テストする組み合わせが増えます。その設定も、ビルドなしに変えられる、本番の一部になります。Hodgsonは、可能なフラグでは、設定をソース管理に置き、再配備を通じて変えるよう勧めます。コードと同じパイプラインを通り、変更をレビューし、記録するためです。運用トグルなど、数分以内に変える必要のあるフラグでこそ、実行時の設定が役立ちます。

実際の活用例

flagdは、OpenFeature仕様に従う、オープンソースのフラグ評価エンジンです。利用者向けの画面を持たず、自分自身では何も保存しません。JSONファイルなどのソースからフラグの定義を読み、サイドカー、アプリのプロセス内、中央のサービスのいずれかとして、リクエストへ答えます。約二十人に一人を新しい購入手続きへ送るフラグは、次のようになります。

```
{
  "$schema": "https://flagd.dev/schema/v0/flags.json",
  "flags": {
    "new-checkout": {
      "state": "ENABLED",
      "variants": {
        "on": true,
        "off": false
      },
      "defaultVariant": "off",
      "targeting": {
        "fractional": [
          ["on", 5],
          ["off", 95]
        ]
      },
      "metadata": {
        "owner": "checkout",
        "expires": "2027-01-31"
      }
    }
  }
}
```

各フラグには、状態と、すべて同じ型でなければならない選択肢の集合があります。既定の選択肢の指定は任意です。`fractional`規則は、相対的な重みで利用者をバケットへ分けるため、5と95なら、約二十人に一人が`on`になります。既定では、フラグのキーと、評価コンテキストの`targetingKey`からバケットを計算します。同じ利用者は常に同じ答えを得ますが、フラグの名前を変えると、全員の振り分けが変わります。`metadata`には、チームがフラグへ付けたい情報を入れます。ここでは担当者と、削除すべき期限です。`state`を`DISABLED`にすると、flagdは値を返さず、呼び出し側が自分の既定値へ戻ります。

そのファイルを使ってflagdを起動し、HTTPで答えを求めます。

```
flagd start --uri file:./flags.json
curl -X POST "localhost:8013/flagd.evaluation.v1.Service/ResolveBoolean" \
  -H "Content-Type: application/json" \
  -d '{"flagKey":"new-checkout","context":{"targetingKey":"user-4711"}}'
```

応答には、値、選択肢、理由の `TARGETING_MATCH` が入ります。アプリでは、フラグのキー、既定値の `false` 、評価コンテキストを、OpenFeatureクライアントへ渡します。トグルポイントが見るのは、真か偽だけです。対象を5%から20%へ広げる変更は、フラグのファイルの編集で、アプリのビルドの変更ではありません。

2025年からの職務で手がけた採用と船員のキャリア用ワークスペースは、記録された実装時点でのリリースフラグの例です。採用側は稼働中で、キャリア側は機能フラグの下にあり、公開が未完了と説明します。これは過去の実装時点の記録で、現在の配備状態を示すものではありません。同じ年からの職務で手がけた運用担当者用の管理画面で、このようなフラグを変えます。技術者が本番データベースを手で編集する代わりに、運用担当者が画面から切り替え、一部の利用者だけへ機能を有効にします。変更は、そこでのほかの操作と同じように、認可し、記録します。

問題が起きるところ

フラグが、いつまでも残ります。 役目を終えたリリースフラグは、負担だけになります。不要な分岐、誰もテストしない経路、誤って切り替えられる設定が残ります。Hodgsonは、保守の負担がかかる在庫として扱い、小さく保つよう勧めます。導入した瞬間に削除タスクをバックログへ入れる、期限を付ける、期限を超えたらテストを失敗させるかアプリの起動を止める仕掛けを置く、総数に上限を設け、一つ加えたら一つ消す、といった対処です。フラグを削除するときは、判定だけでなく、選ばれなくなった経路も消します。

古いフラグに、新しい意味を与えます。 2012年8月1日、Knight Capitalの自動注文ルーター、SMARSは、45分間に何百万もの注文を市場へ送り、同社は4億6,000万ドルを超える損失を出しました。SECの命令書が、経緯を示します。New York Stock ExchangeのRetail Liquidity Program用の新しいコードは、受信する注文に含まれるフラグを再利用しました。かつては、2003年に使用をやめたものの削除していなかったPower Pegという機能を、有効にするフラグでした。配備から八台中一台が漏れ、そのサーバーでは、古い機能が動き出しました。その後、担当者がほかの七台から新しいコードを消すと、そこでもPower

Pegが有効になりました。使い終えたフラグへ新しい意味を与えず、守っていた不要なコードも一緒に消し、有効にする前に、全サーバーが同じビルドで動くか確かめます。

誰も試していない組み合わせがあります。 二つのフラグで四通り、十個で千通りを超えます。Hodgsonは、組み合わせの爆発と呼びます。全通りをテストできることは、めったにありません。本番へ届く組み合わせを試します。現在の本番設定で、公開予定のフラグを有効にしたもの、それを無効にした代替設定、そして多くのチームが加える、すべて有効の設定です。無効は常に既存の振る舞い、有効は新しい振る舞い、と決めておくと役立ちます。

既定値を、誰も選んでいません。 OpenFeature仕様は、フラグの評価で例外を投げてはならないとします。問題があれば、呼び出し側が渡した既定値を返し、詳細な結果には、エラーコードが入ります。クライアントは、ログにも書くべきではありません。フラグの定義より先に配備したアプリが、`FLAG_NOT_FOUND` でログを埋めるのを避けるためです。アプリは動き続けますが、キーを打ち間違えると、全利用者へ、気付かれないまま既定値が返ることにもなります。既存の安全な振る舞いを既定値として渡し、仕様がログの記録場所とするフックを付けます。エラーで終わる評価を数え、発生したらアラートを出します。

誰でも設定できる上書きがあります。 特別なCookie、URLのパラメーター、HTTPヘッダーで、テスト担当者がフラグを強制的に有効にできると便利です。Hodgsonは、好奇心の強い利用者や悪意のある利用者も同じ上書きを使い、自分のフラグ状態を変えられると指摘します。内部のトラフィックだけで受け付けるか、暗号学的に署名します。懸念は和らぎますが、Hodgsonが述べるように、複雑さと攻撃面も増えます。

参考資料

- 機能トグル、別名機能フラグ — Martin FowlerのサイトにあるPete Hodgsonの記事です。トグルの構成、四分類、テスト、在庫を小さく保つ方法を扱います。
- フラグ評価API — 既定値、エラーコード、評価で例外を投げない理由を含む、OpenFeatureの評価仕様です。
- flagd：フラグの定義 — flagdが読むJSON形式の、状態、選択肢、既定の選択肢、対象の指定、メタデータです。
- flagd：割合による操作 — 決定論的な割合の分割と、利用者が入るバケットを決めるものです。
- Knight Capital Americas LLCに関する件 — 再利用したフラグを含む、2012年8月の取引損失についての、SECの2013年の命令書です。

用語集

実際の制約に合わせて選ぶ

業務の流れ、既存のインターフェース、変更を必要とする利用者をお知らせください。検証可能な納品内容と、お客様のチームが保守を続けられる引き継ぎ内容を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「機能フラグとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/feature-flag/