Engineer

Прапорець функції

⚑ вмикає й вимикає код без повторного збирання застосунку

Що це таке

Прапорець функції — це точка в коді, де рішення ухвалюється на основі налаштування, яке команда може змінити, не збираючи застосунок заново. За прапорцем стоїть новий шлях виконання, який уже працює в продакшні, але лишається неактивним, доки налаштування не ввімкне його для всіх або для обраної групи користувачів. Піт Ходжсон докладно описав цей прийом на сайті Мартіна Фаулера: статтю публікували частинами з січня 2016 року, а в жовтні 2017 року автор її доопрацював. Він називає такі прапорці feature toggles, а серед інших назв того самого згадує feature flags, feature bits і feature flippers.

Ходжсон розкладає прапорець на частини. Toggle point — це місце в коді, яке питає, чи ввімкнено новий шлях. Toggle router відповідає на це питання, спираючись на конфігурацію прапорців, а коли відповідь залежить від того, хто питає, — ще й на toggle context: зазвичай це поточний користувач або спеціальний файл cookie чи HTTP-заголовок, що його заступає. Ходжсон застерігає від того, щоб зв’язувати ці дві частини надто тісно. Код у toggle point має ставити вузьке питання, наприклад, чи містить це підтвердження замовлення нове посилання для скасування, а правило, яке це вирішує, живе в одному місці за toggle router, тож його можна змінити, не чіпаючи кожне місце виклику.

Прапорці різняться тим, скільки вони живуть і як часто змінюється їхня відповідь, і Ходжсон ділить їх на чотири види. Release toggles дають змогу відправити в продакшн незавершений і неперевірений код як латентний, тобто такий, що поки не виконується; за словами Ходжсона, зазвичай вони не повинні затримуватися в коді набагато довше, ніж тиждень-два. Experiment toggles ділять користувачів на групи для A/B-тесту, причому кожен користувач щоразу потрапляє в ту саму групу. Ops toggles дають змогу тим, хто експлуатує систему, змінювати її поведінку в продакшні. Багато з них

живуть недовго, але кілька довгожителів, аварійні вимикачі (kill switches), дозволяють згорнути функціональність, без якої система може обійтися, коли навантаження незвично високе, наприклад панель рекомендацій, яку дорого генерувати; Ходжсон порівнює їх із запобіжником (circuit breaker), яким керують уручну. Permissioning toggles визначають, які функції бачить конкретний користувач, наприклад преміальні функції для тих, хто платить, або альфа-функції для власних працівників компанії, і можуть існувати роками.

Прапорець функції — це не канарковий реліз, хоча вони часто працюють у парі. Канарковий реліз тримає дві версії сервісу поруч і спрямовує частку трафіку на нову; прапорець постачає одну версію, що містить обидва шляхи, і обирає між ними під час кожного запиту. Довгоживуча гілка в системі контролю версій теж тримає нову роботу окремо, але тоді код потрапляє в продакшн лише після злиття гілки, і то весь одразу.

Застосуйте поняття до рішення

Застосуйте «Що таке прапорець функції?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Одна команда веде ідею від ескізу до продукту, у який ваші клієнти вже можуть увійти, а не перекидає роботу між трьома підрядниками.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Збільшила дохід GIS-застосунку в 10 разів — 10x зростання виторгу завдяки картографічному продукту

Збільшила дохід GIS-застосунку в 10 разів Зв'язатися з нами →

Чому це важливо

Головна причина використовувати прапорці — відокремити розгортання від релізу. Ходжсон описує release toggles як найпоширеніший спосіб утілити на практиці один із принципів Continuous Delivery: реліз функції — це окремий крок, не те саме, що розгортання її коду. Код може щодня потрапляти в продакшн з однієї спільної основної гілки, а функція, над якою працюють місяць, лишається вимкненою, доки не буде готова, і саме це робить розробку в основній гілці (trunk-based development) здійсненною.

Оскільки прапорець може давати різним користувачам різні відповіді, реліз можна розширювати поетапно. Ходжсон описує, як нові функції спершу вмикають для внутрішніх користувачів; він називає це Champagne Brunch, бо компанія першою п’є власне шампанське. Далі йде канаркова група: у його

прикладі це випадковий 1 відсоток користувачів, відібраний за остачею від ділення ідентифікатора користувача, щоб у групі лишалися ті самі люди, тоді як решта 99 відсотків зберігає стару поведінку. Якщо ця група натрапляє на проблеми, прапорець вимикають, і нікому не доводиться відкочувати розгортання.

Усе це має свою ціну. Кожен прапорець додає в код ще одне розгалуження й ще одну комбінацію, яку треба перевірити, а конфігурація за ним стає частиною продакшну, яку можна змінити без нової збірки. Ходжсон радить тримати конфігурацію прапорців у системі контролю версій і змінювати її через повторні розгортання всюди, де прапорець це дозволяє, щоб конфігурація проходила той самий конвеєр, що й код, а зміни в ній переглядали й фіксували. Конфігурація, яку змінюють просто під час роботи системи, виправдовує себе там, де прапорець мусить перемикатися за лічені хвилини, як-от в ops toggles.

Як це виглядає на практиці

flagd — це рушій обчислення прапорців із відкритим кодом, що відповідає специфікації OpenFeature. Він не має інтерфейсу користувача й нічого не зберігає сам: він читає визначення прапорців із JSON-файлу або іншого джерела й відповідає на запити — як sidecar, усередині процесу застосунку або як центральний сервіс. Прапорець, що спрямовує на нове оформлення замовлення приблизно кожного двадцятого користувача, виглядає так:

```
{
  "$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"
```

```
      }
    }
  }
}
```

Кожен прапорець має стан і набір варіантів, які мають бути одного типу; варіант за замовчуванням необов'язковий. Правило `fractional` розподіляє користувачів по сегментах за відносною вагою, тож 5 і 95 спрямовують на `on` приблизно кожного двадцятого. За замовчуванням сегмент обчислюється з ключа прапорця та `targetingKey` у контексті обчислення, а отже, той самий користувач завжди отримує ту саму відповідь, а перейменування прапорця перемішує всіх. У `metadata` лежить усе, що команда хоче прив'язати до прапорця, тут — власник і дата, до якої прапорець має зникнути. Якщо задати `state` значення `DISABLED`, flagd узагалі не поверне значення, і код, що до нього звертається, скористається власним значенням за замовчуванням.

Запустіть flagd із цим файлом і попросіть відповідь через HTTP:

```
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"}}'
```

Відповідь містить значення, варіант і причину `TARGETING_MATCH`. У застосунку виклик іде через клієнт OpenFeature з ключем прапорця, значенням за замовчуванням `false` і контекстом обчислення, і toggle point не бачить нічого, крім булевого значення. Щоб збільшити частку з 5 до 20 відсотків, досить змінити файл прапорців, а не збірку застосунку.

Робочий простір найму та кар'єрний робочий простір моряків 2025 року показують прапорець релізу в зафіксованому стані реалізації: в описі робочий простір найму названо запущеним, а кар'єрний робочий простір — керованим прапорцем функції та випущеним не повністю. Це історичний знімок стану, а не твердження про поточний стан розгортання. Операторський бек-офіс того самого року — це місце, де такі прапорці змінюють. Оператори перемикають прапорці й вмикають функцію лише для частини користувачів з екрана інтерфейсу замість того, щоб інженер вручну редагував виробничу базу даних, і кожну зміну авторизують і фіксують так само, як будь-яку іншу дію там.

Де виникають проблеми

Прапорці, які ніколи не зникають. Прапорець релізу, що вже виконав свою роботу, — це чисті витрати: розгалуження, яке нікому не потрібне, шлях, який ніхто не тестує, і налаштування, яке хтось може перемкнути помилково. Ходжсон пропонує ставитися до прапорців як до запасів, що мають ціну, і тримати ці запаси малими. Серед його засобів — завдання на вилучення в беклозі з тієї миті, як з'являється прапорець релізу, термін придатності, часова бомба, яка валить тест або не дає застосунку запуститися, коли прапорець пережив свою дату, і фіксована межа кількості прапорців, щоб новий можна було додати, лише вилучивши інший. Вилучити прапорець означає видалити не лише перевірку, а й той шлях коду, що програв.

Старий прапорець із новим значенням. 1 серпня 2012 року автоматизований маршрутизатор ордерів Knight Capital, SMARS, за 45 хвилин відправив на ринок мільйони ордерів, і компанія втратила понад 460 мільйонів доларів. Постанова Комісії з цінних паперів і бірж США (SEC) описує, як це сталося. Новий код для програми Retail Liquidity Program Нью-Йоркської фондової біржі повторно використав прапорець, який передавався разом із вхідними ордерами й колись умикав Power Peg — функцію, якою Knight перестала користуватися у 2003 році, але так і не видалила. Технік не скопіював новий код на один із восьми серверів, тож на цьому сервері повторно використаний прапорець розбудив стару функцію. Коли співробітники потім прибрали новий код із решти семи, прапорець активував Power Peg і там. Ніколи не надавайте відслуженому прапорцю нового значення, видаляйте мертвий код разом із прапорцем, що його охороняв, і перевіряйте, що на всіх серверах працює та сама збірка, перш ніж вмикати прапорець.

Комбінації, яких ніхто не перевіряв. Два прапорці дають чотири комбінації, а десять — понад тисячу; Ходжсон називає це комбінаторним вибухом. Перевірити всі рідко вдається, тож перевіряйте ті, що можуть потрапити в продакшн: поточну конфігурацію продакшну з увімкненими прапорцями, які ось-ось вийдуть у реліз, запасну конфігурацію з тими самими прапорцями вимкненими і, як роблять також багато команд, усе ввімкнене. Допомагає стала домовленість: вимкнено — це завжди наявна поведінка, увімкнено — нова.

Значення за замовчуванням, якого ніхто не обирав. Специфікація OpenFeature каже, що обчислення прапорця не повинно кидати виняток. Коли щось іде не так, виклик повертає значення за замовчуванням, яке

передав той, хто викликає, детальна форма результату містить код помилки, а клієнтові не слід ще й писати в журнал, бо інакше застосунок, розгорнутий раніше, ніж визначено його прапорець, засипав би журнал повідомленнями `FLAG_NOT_FOUND`. Так застосунок працює далі, але це також означає, що одна описка в ключі прапорця непомітно перетворюється на значення за замовчуванням для всіх користувачів. Передавайте як значення за замовчуванням стару, безпечну поведінку й підключіть до обчислення обробник, який рахує помилки й здіймає тривогу, щойно вони з'являються; саме в такому обробнику специфікація й пропонує вести журналювання.

Перевизначення, які може задати будь-хто. Зручно дати тестувальникам змогу примусово вмикати прапорець спеціальним файлом cookie, параметром в URL або HTTP-заголовком. Ходжсон зауважує, що допитливі чи зловмисні користувачі можуть задати те саме перевизначення й змінити стан прапорця для себе. Приймайте перевизначення лише з внутрішнього трафіку або підписуйте їх криптографічно; це зменшує ризик, але, як зазначає Ходжсон, додає власної складності й поверхні атаки.

Що почитати далі

- Feature Toggles (aka Feature Flags) — стаття Піта Ходжсона на сайті Мартіна Фаулера: частини прапорця, чотири категорії, тестування і як тримати запас прапорців малим.
- Flag Evaluation API — специфікація OpenFeature щодо обчислення прапорця, зокрема значення за замовчуванням, коди помилок і чому обчислення ніколи не кидає виняток.
- flagd: Flag Definitions — JSON-формат, який читає flagd: стан, варіанти, варіант за замовчуванням, таргетинг і метадані.
- flagd: Fractional Operation — детерміновані відсоткові розподіли і що визначає, до якого сегмента потрапить користувач.
- In the Matter of Knight Capital Americas LLC — постанова SEC 2013 року про торгові збитки в серпні 2012 року, зокрема про повторно використаний прапорець.

Словник

Обирайте за фактичними умовами

Опишіть робочий процес, наявні інтерфейси та користувачів, яким потрібна зміна. Ми можемо визначити перевірюваний результат і спосіб передачі, який дасть вашій команді змогу підтримувати його.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке прапорець функції?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/feature-flag/