Engineer

Feature flag

⚑ en kontakt, der slår kode til og fra uden at genbygge appen

Hvad det er

Et feature flag er et beslutningspunkt i koden, der læser en indstilling, som teamet kan ændre uden at bygge applikationen igen. Bag flaget ligger en ny kodesti, der allerede er i produktion, men som ligger i dvale, indtil indstillingen slår den til, for alle eller for en udvalgt gruppe brugere. Pete Hodgson beskrev teknikken grundigt på Martin Fowlers website i flere afsnit fra januar 2016 og reviderede teksten i oktober 2017. Han bruger betegnelsen feature toggles og nævner feature flags, feature bits og feature flippers som andre navne for samme ting.

Hodgson skiller et flag ad i flere dele. Et toggle point er det sted i koden, der spørger, om den nye sti er slået til. En toggle router besvarer spørgsmålet ud fra toggle-konfigurationen og, når svaret afhænger af, hvem der spørger, ud fra en toggle context: som regel den aktuelle bruger eller en særlig cookie eller HTTP-header, der træder i stedet for brugeren. Han advarer mod at koble de to sammen. Koden ved et toggle point bør stille et snævert spørgsmål, for eksempel om denne ordrebekræftelse skal have det nye link til annullering, mens reglen, der afgør det, ligger ét sted bag routeren, så den kan ændres uden at røre ved hvert eneste kaldested.

Flag adskiller sig i, hvor længe de lever, og hvor ofte deres svar skifter, og Hodgson sorterer dem i fire slags. Release toggles lader ufærdig og utestet kode gå i produktion som latent kode; ifølge Hodgson bør de som regel ikke blive siddende ret meget længere end en uge eller to. Experiment toggles deler brugerne i grupper til en A/B-test, så hver bruger havner i samme gruppe hver gang. Ops toggles lader driftsfolkene ændre, hvordan systemet opfører sig i produktion. Mange af dem er kortlivede, men nogle få langlivede kill switches lader driftsfolkene skrue ned for funktioner, systemet kan undvære, når belastningen er usædvanligt høj, for eksempel et anbefalingspanel, der er dyrt at generere; Hodgson sammenligner dem med en manuelt styret circuit breaker.

Permissioning toggles afgør, hvilke funktioner en bestemt bruger ser, for eksempel premiumfunktioner for betalende kunder eller alfafunktioner for virksomhedens egne medarbejdere, og de kan blive i årevis.

Et feature flag er ikke en canary release, selvom de to ofte arbejder sammen. En canary kører to versioner af en tjeneste side om side og sender en andel af trafikken til den nye; et flag sender én version ud, der indeholder begge stier, og vælger mellem dem ved hver forespørgsel. En langlivet branch i versionsstyringen holder også nyt arbejde adskilt, men så når koden først produktion, når branchen bliver merget, og så på én gang.

Brug begrebet i en beslutning

Brug »Hvad er et feature flag?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Ét team, der tager en idé fra skitse til noget, jeres kunder kan logge ind på, i stedet for at kaste den mellem tre leverandører.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har ført en GIS-kortapp til 10x omsætningsvækst — 10x omsætningsvækst, drevet af kortproduktet

Har ført en GIS-kortapp til 10x omsætningsvækst Kontakt os →

Hvorfor det betyder noget

Den vigtigste grund til at bruge flag er at adskille udrulning fra frigivelse. Hodgson beskriver release toggles som den mest almindelige måde at omsætte et princip fra Continuous Delivery til praksis: Frigivelsen af en funktion er et særskilt skridt, adskilt fra udrulningen af dens kode. Kode kan gå i produktion hver dag fra én fælles hovedbranch, mens en funktion, der tager en måned at bygge, forbliver slukket, indtil den er klar, og det er det, der gør trunk-baseret udvikling mulig i praksis.

Fordi et flag kan give forskellige svar til forskellige brugere, kan en frigivelse udvides i trin. Hodgson beskriver, hvordan nye funktioner først slås til for interne brugere, hvad han kalder en Champagne Brunch, fordi virksomheden selv får lov at drikke sin champagne først. Derefter kommer en canary-gruppe: i hans eksempel en tilfældig gruppe på 1 procent af brugerne, udvalgt med en modulo af bruger-ID'et, så de samme personer bliver i gruppen, mens de øvrige 99 procent beholder den gamle adfærd. Melder den gruppe om problemer, slås flaget fra, og ingen behøver at rulle en deployment tilbage.

Det har sin pris. Hvert flag tilføjer en forgrening i koden og en kombination, der skal testes, og konfigurationen bag det bliver en del af produktionen, som kan ændres uden et nyt build. Hodgson råder til at holde toggle-konfigurationen i versionsstyring og ændre den gennem nye udrulninger, hvor flaget tillader det, så konfigurationen går gennem den samme pipeline som koden, og ændringer i den bliver gennemgået og registreret. Det er ved flag, der skal kunne skifte inden for minutter, som ops toggles, at konfiguration under kørsel betaler sig.

Sådan ser det ud i praksis

flagd er en open source-motor til evaluering af flag, der følger OpenFeature-specifikationen. Den har ingen brugerflade og gemmer intet selv; den læser flagdefinitioner fra en JSON-fil eller en anden kilde og besvarer forespørgsler, enten som sidecar, inde i applikationens proces eller som en central tjeneste. Et flag, der sender cirka hver tyvende bruger til et nyt checkout-flow, ser sådan ud:

```
{
  "$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"
      }
    }
  }
}
```

Hvert flag har en tilstand og et sæt varianter, der alle skal være af samme type; standardvarianten er valgfri. Reglen `fractional` fordeler brugerne i grupper efter relativ vægt, så 5 og 95 sender cirka hver tyvende bruger til `on`. Som standard afgøres det, hvilken gruppe en bruger havner i, ud fra flagets nøgle og

`targetingKey` i evalueringskonteksten. Den samme bruger får derfor altid det samme svar, og alle bliver blandet om, hvis flaget skifter navn. `metadata` rummer det, teamet vil knytte til flaget, her en ejer og den dato, hvor flaget skal være væk. Sætter man `state` til `DISABLED`, svarer flagd slet ikke med en værdi, og den, der kalder, falder tilbage på sin egen standardværdi.

Kør flagd mod filen, og spørg den om et svar over 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"}}'
```

Svaret indeholder værdien, varianten og årsagen `TARGETING_MATCH`. I applikationen går kaldet gennem en OpenFeature-klient med flagets nøgle, standardværdien `false` og evalueringskonteksten, og et toggle point ser intet andet end sandt eller falsk. At hæve andelen fra 5 til 20 procent er en ændring i flagfilen, ikke i applikationens build.

Arbejdsområderne til rekruttering og søfarendes karriere fra 2025 viser et release-flag i det dokumenterede øjebliksbillede af implementeringen: det beskriver rekrutteringsdelen som live og karriere-workspacet som liggende bag et feature flag uden at være fuldt frigivet. Det er et historisk øjebliksbillede, ikke en erklæring om den aktuelle driftstilstand. Operatørernes back-office fra samme år er stedet, hvor den slags flag bliver ændret. Operatørerne slår flag til og fra og slår en funktion til for kun en del af brugerne fra et skærmbillede, i stedet for at en udvikler retter i produktionsdatabasen i hånden, og hver ændring bliver autoriseret og registreret ligesom enhver anden handling dér.

Hvor det går galt

Flag, der aldrig forsvinder. Et release-flag, der har gjort sit arbejde, er ren omkostning: en forgrening, ingen har brug for, en sti, ingen tester, og en indstilling, nogen kan komme til at slå om ved en fejl. Hodgson foreslår at betragte flag som et lager, der koster noget, og at holde lageret lille. Blandt hans midler er en opgave om flagets fjernelse i backloggen allerede ved indførelsen af et release-flag, en udløbsdato, en tidsbombe, der får en test til at fejle eller

forhindrer applikationen i at starte, når et flag har overlevet sin dato, og et fast loft over antallet af flag, så et nyt flag kræver, at et andet bliver fjernet. At fjerne et flag betyder at slette den kodesti, der tabte, og ikke kun tjekket.

Et gammelt flag med en ny betydning. Den 1. august 2012 sendte Knight Capitals automatiske ordrerouter, SMARS, millioner af ordrer ud på markedet i løbet af 45 minutter, og firmaet tabte mere end 460 millioner dollar. SEC's afgørelse beskriver, hvordan det skete. Ny kode til New York Stock Exchanges Retail Liquidity Program genbrugte et flag, der fulgte med indkommende ordrer, og som engang havde slået Power Peg til, en funktion, Knight var holdt op med at bruge i 2003, men aldrig havde fjernet. En tekniker fik ikke kopieret den nye kode til én af de otte servere, så på den server vækkede det genbrugte flag den gamle funktion. Da medarbejderne derefter fjernede den nye kode fra de syv andre, aktiverede flaget også Power Peg dér. Giv aldrig et udtjent flag en ny betydning, slet død kode sammen med det flag, der beskyttede den, og tjek, at alle servere kører samme build, før et flag slås til.

Kombinationer, ingen har testet. To flag giver fire kombinationer, og ti giver over tusind; Hodgson kalder det en kombinatorisk eksplosion. Det er sjældent muligt at teste dem alle, så test dem, der kan nå produktion: den nuværende produktionskonfiguration med de flag, der er ved at blive frigivet, slået til, reservekonfigurationen med de samme flag slået fra og, som mange teams også gør, alt slået til. En fast konvention hjælper: Fra betyder altid den eksisterende adfærd, og til betyder den nye.

En standardværdi, ingen valgte. OpenFeature-specifikationen siger, at evaluering af et flag ikke må kaste en exception. Går noget galt, returnerer kaldet den standardværdi, kalderen har givet med, den detaljerede form af resultatet indeholder en fejlkode, og klienten bør heller ikke skrive til loggen, for ellers ville en applikation, der er sendt i produktion, før flaget er defineret, fylde loggen med `FLAG_NOT_FOUND`. Det holder applikationen kørende, men det betyder også, at en tastefejl i en flagnøgle i al stilhed bliver til standardværdien for alle brugere. Giv den gamle, sikre adfærd med som standardværdi, og kobl en udvidelse på evalueringen, der tæller fejl og slår alarm, når de dukker op; det er også i sådan en udvidelse, specifikationen placerer logning.

Overrides, som alle kan sætte. Det er bekvemt at lade testere tvinge et flag til med en særlig cookie, en parameter i URL'en eller en HTTP-header. Hodgson påpeger, at nysgerrige eller ondsindede brugere kan sætte den samme override

og ændre flagets tilstand for sig selv. Acceptér kun overrides fra intern trafik, eller signér dem kryptografisk; det mindsker bekymringen, men bringer, som Hodgson bemærker, sin egen kompleksitet og angrebsflade med sig.

Læs mere

- Feature Toggles (aka Feature Flags) — Pete Hodgsons artikel på Martin Fowlers website: et flags dele, de fire kategorier, test, og hvordan man holder lageret af flag lille.
- Flag Evaluation API — OpenFeature-specifikationen for evaluering af et flag, herunder standardværdier, fejlkoder, og hvorfor evalueringen aldrig kaster en exception.
- flagd: Flag Definitions — det JSON-format, flagd læser: tilstand, varianter, standardvariant, targeting og metadata.
- flagd: Fractional Operation — deterministiske procentvise opdelinger, og hvad der afgør, hvilken gruppe en bruger havner i.
- In the Matter of Knight Capital Americas LLC — SEC's afgørelse fra 2013 om handelstabet i august 2012, herunder det genbrugte flag.

Ordliste

Vælg ud fra jeres faktiske vilkår

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er et feature flag?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/feature-flag/