Engineer

Funksjonsflagg

⚑ en av/på-bryter for kode uten å bygge applikasjonen på nytt

Hva det er

Et funksjonsflagg er et beslutningspunkt i koden som leser en innstilling teamet kan endre uten å bygge applikasjonen på nytt. Bak flagget ligger en ny kodevei som allerede er i produksjon, men som ligger i dvale til innstillingen slår den på, for alle eller for en utvalgt gruppe brukere. Pete Hodgson beskrev teknikken grundig på Martin Fowlers nettsted, i deler fra januar 2016 og med en revisjon i oktober 2017. Han kaller dem feature toggles og nevner feature flags, feature bits og feature flippers som andre navn på det samme.

Hodgson deler et flagg opp i bestanddeler. Toggle point er stedet i koden som spør om den nye veien er slått på. Toggle router svarer på spørsmålet ut fra konfigurasjonen av flaggene og, når svaret avhenger av hvem som spør, ut fra en toggle context: som regel den aktuelle brukeren, eller en spesiell cookie eller HTTP-header som står i stedet for brukeren. Han advarer mot å koble de to for tett sammen. Koden i toggle point bør stille et smalt spørsmål, for eksempel om denne ordrebekreftelsen skal ha den nye lenken for kansellering, mens regelen som avgjør det, bor på ett sted bak routeren, slik at den kan endres uten å røre hvert sted i koden der spørsmålet stilles.

Flagg skiller seg fra hverandre i hvor lenge de lever og hvor ofte svaret deres endres, og Hodgson sorterer dem i fire typer. Release toggles lar uferdig, utestet kode gå i produksjon som latent kode, altså kode som ennå ikke kjøres; ifølge Hodgson bør de som regel ikke bli værende stort lenger enn en uke eller to. Experiment toggles deler brukerne i grupper for en A/B-test, der hver bruker havner i samme gruppe hver gang. Ops toggles lar de som drifter systemet, endre hvordan det oppfører seg i produksjon. Mange av dem er kortlivede, men noen få langlivede nødbrytere (kill switches) lar driften skru ned funksjonalitet systemet klarer seg uten når belastningen er uvanlig høy, for eksempel et anbefalingspanel som er dyrt å generere; Hodgson sammenligner dem med en

manuelt styrt circuit breaker. Permissioning toggles avgjør hvilke funksjoner en bestemt bruker ser, for eksempel premiumfunksjoner for betalende kunder eller alfafunksjoner for selskapets egne ansatte, og de kan bli værende i årevis.

Et funksjonsflagg er ikke en canary release, selv om de to ofte brukes sammen. En canary kjører to versjoner av en tjeneste side om side og sender en andel av trafikken til den nye; et flagg leverer én versjon som inneholder begge veiene, og velger mellom dem for hver forespørsel. En langlivet gren i versjonskontrollen holder også nytt arbeid adskilt, men da kommer koden i produksjon først når grenen flettes inn, og da alt på én gang.

Bruk begrepet i en beslutning

Bruk «Hva er et funksjonsflagg?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Ett team som tar en idé fra skisse til noe kundene dine kan logge inn på, i stedet for at den sendes videre mellom tre leverandører.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Ledet en GIS-kartapp til 10 ganger mer omsetning — 10x inntektsvekst, ledet fra kartproduktet

Ledet en GIS-kartapp til 10 ganger mer omsetning Ta kontakt →

Hvorfor det er viktig

Hovedgrunnen til å bruke flagg er å skille produksjonssetting fra lansering. Hodgson beskriver release toggles som den vanligste måten å sette et prinsipp fra Continuous Delivery ut i livet på: lanseringen av en funksjon er et eget steg, atskilt fra produksjonssettingen av koden. Kode kan gå i produksjon hver dag fra én felles hovedgren, mens en funksjon som det tar en måned å bygge, holdes skjult til den er klar, og det er det som gjør trunk-basert utvikling (trunk-based development) gjennomførbar.

Fordi et flagg kan gi ulike svar til ulike brukere, kan en lansering utvides i trinn. Hodgson beskriver hvordan nye funksjoner først slås på for interne brukere, noe han kaller en Champagne Brunch, siden selskapet får drikke sin egen champagne først. Deretter kommer en canary-gruppe: i hans eksempel tilfeldig utvalgte 1 prosent av brukerne, valgt med modulo av bruker-ID-en, slik at de samme menneskene blir i gruppen, mens de øvrige 99 prosentene beholder den gamle oppførselen. Melder den gruppen om problemer, slås flagget av, og ingen trenger å rulle tilbake en produksjonssetting.

Alt dette har en pris. Hvert flagg legger til en forgrening i koden og en kombinasjon som må testes, og konfigurasjonen bak det blir en del av produksjon som kan endres uten et nytt bygg. Hodgson råder til å holde konfigurasjonen av flaggene i versjonskontroll og endre den gjennom nye produksjonssettinger der flagget tillater det, slik at konfigurasjonen går gjennom den samme pipelinen som koden, og endringene blir gjennomgått og registrert. Konfigurasjon som endres mens systemet kjører, gjør nytte for seg der et flagg må kunne endres i løpet av minutter, som ops toggles.

Slik ser det ut i praksis

flagd er en motor med åpen kildekode for evaluering av flagg, og den følger OpenFeature-spesifikasjonen. Den har ikke noe brukergrensesnitt og lagrer ingenting selv; den leser flaggdefinisjoner fra en JSON-fil eller en annen kilde og svarer på forespørsler, enten som sidecar, inne i applikasjonsprosessen eller som en sentral tjeneste. Et flagg som sender omtrent hver tjuende bruker til en ny kasseløsning, ser slik ut:

```
{
  "$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 flagg har en tilstand og et sett varianter som alle må være av samme type; standardvarianten er valgfri. Regelen `fractional` fordeler brukerne i bøtter etter relativ vekt, så 5 og 95 sender omtrent hver tjuende bruker til `on`. Som standard regnes bøtta ut fra flaggnøkkelen og `targetingKey` i evalueringskonteksten, noe som betyr at samme bruker alltid får samme svar, og at et nytt navn på flagget stokker om alle. `metadata` rommer det teamet vil knytte til flagget, her en eier og datoen da flagget skal være borte. Settes `state` til `DISABLED`, svarer flagd uten noen verdi i det hele tatt, og koden som spør, faller tilbake på sin egen standardverdi.

Kjør flagd mot filen og be 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 inneholder verdien, varianten og grunnen `TARGETING_MATCH`. I applikasjonen går kallet gjennom en OpenFeature-klient med flaggnøkkelen, standardverdien `false` og evalueringskonteksten, og toggle point ser ikke annet enn sant eller usant. Å øke andelen fra 5 til 20 prosent er en endring i flaggfilen, ikke i bygget av applikasjonen.

Arbeidsområdet for rekruttering og karrierearbeidsområdet for sjøfolk, fra 2025, viser et lanseringsflagg i det dokumenterte øyeblikksbildet av implementeringen: der beskrives arbeidsområdet for rekruttering som satt i drift og karrierearbeidsområdet for sjøfolk som styrt av et funksjonsflagg og ikke fullt lansert. Det er et historisk øyeblikksbilde, ikke en erklæring om den nåværende produksjonstilstanden. Operatørenes backoffice fra samme år er stedet der slike flagg blir endret. Operatørene slår flagg av og på og aktiverer en funksjon for bare noen brukere fra et skjermbilde, i stedet for at en ingeniør redigerer produksjonsdatabasen for hånd. Hver endring blir autorisert og registrert som enhver annen handling der.

Hvor det går galt

Flagg som aldri forsvinner. Et release-flagg som har gjort jobben sin, er ren kostnad: en forgrening ingen trenger, en vei ingen tester, og en innstilling noen kan komme til å slå om ved en feil. Hodgson foreslår å behandle flagg som en

lagerbeholdning som koster noe, og å holde den beholdningen liten. Blant tiltakene hans er en oppgave om å fjerne flagget, opprettet i backloggen i det øyeblikket et release-flagg innføres, en utløpsdato, en tidsbombe som får en test til å feile eller hindrer applikasjonen i å starte når et flagg har levd forbi datoen sin, og en fast grense for antall flagg, slik at et nytt bare kan legges til når et annet fjernes. Å fjerne et flagg betyr å slette den tapende kodeveien i tillegg til selve sjekken.

Et gammelt flagg med ny betydning. 1. august 2012 sendte Knight Capitals automatiserte ordreruter, SMARS, millioner av ordrer ut i markedet i løpet av 45 minutter, og selskapet tapte mer enn 460 millioner dollar. Vedtaket fra det amerikanske tilsynet SEC beskriver hvordan det skjedde. Ny kode for New York-børsens Retail Liquidity Program gjenbrukte et flagg som fulgte med innkommende ordrer, og som en gang hadde slått på Power Peg, en funksjon Knight hadde sluttet å bruke i 2003, men aldri fjernet. En tekniker kopierte ikke den nye koden til én av de åtte serverne, så på den serveren vekket det gjenbrukte flagget den gamle funksjonen. Da de ansatte deretter fjernet den nye koden fra de sju andre, aktiverte flagget Power Peg der også. Gi aldri et pensjonert flagg en ny betydning, slett død kode sammen med flagget som voktet den, og kontroller at alle servere kjører det samme bygget før et flagg slås på.

Kombinasjoner ingen har testet. To flagg gir fire kombinasjoner, og ti gir over tusen; Hodgson kaller det en kombinatorisk eksplosjon. Å teste alle er sjelden mulig, så test dem som kan nå produksjon: den gjeldende produksjonskonfigurasjonen med flaggene som snart skal lanseres, slått på, reservekonfigurasjonen med de samme flaggene slått av og, slik mange team også gjør, alt slått på. En fast konvensjon hjelper: av betyr alltid den eksisterende oppførselen, og på betyr den nye.

En standardverdi ingen valgte. OpenFeature-spesifikasjonen sier at evaluering av et flagg ikke skal kaste unntak. Når noe går galt, returnerer kallet den standardverdien kalleren sendte med, den detaljerte formen av resultatet har en feilkode, og klienten bør heller ikke skrive til loggen, fordi en applikasjon som ble satt i produksjon før flagget var definert, ellers ville fylt loggen med `FLAG_NOT_FOUND`. Det holder applikasjonen i gang, men det betyr også at en skrivefeil i en flaggnøkkel i det stille blir standardverdien for alle brukere. Send med den gamle, trygge oppførselen som standardverdi, og koble en utvidelse på evalueringen som teller feil og slår alarm når de dukker opp; det er også i en slik utvidelse spesifikasjonen plasserer logging.

Overstyringer hvem som helst kan sette. Det er praktisk å la testere tvinge et flagg på med en spesiell cookie, en parameter i URL-en eller en HTTP-header. Hodgson påpeker at nysgjerrige eller ondsinnede brukere kan sette den samme overstyringen og endre tilstanden til et flagg for seg selv. Godta overstyringer bare fra intern trafikk, eller signer dem kryptografisk; det reduserer risikoen, men legger, som Hodgson bemerker, til egen kompleksitet og angrepsflate.

Les mer

- Feature Toggles (aka Feature Flags) — Pete Hodgsons artikkel på Martin Fowlers nettsted: delene i et flagg, de fire kategoriene, testing og hvordan man holder beholdningen av flagg liten.
- Flag Evaluation API — OpenFeature-spesifikasjonen for evaluering av et flagg, blant annet standardverdier, feilkoder og hvorfor evalueringen aldri kaster unntak.
- flagd: Flag Definitions — JSON-formatet flagd leser: tilstand, varianter, standardvariant, målretting og metadata.
- flagd: Fractional Operation — deterministiske prosentvise fordelinger og hva som avgjør hvilken bøtte en bruker havner i.
- In the Matter of Knight Capital Americas LLC — SEC-vedtaket fra 2013 om handelstapet i august 2012, blant annet det gjenbrukte flagget.

Ordliste

Velg ut fra deres faktiske rammer

Beskriv arbeidsflyten, eksisterende grensesnitt og brukerne som trenger endringen. Vi kan definere en testbar leveranse og en overlevering som gjør teamet i stand til å vedlikeholde løsningen.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er et funksjonsflagg?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/feature-flag/