Engineer

Funktionsflagga

⚑ en av/på-brytare för kod utan att bygga om applikationen

Vad det är

En funktionsflagga är en beslutspunkt i koden som läser en inställning som teamet kan ändra utan att bygga applikationen på nytt. Bakom flaggan ligger en ny kodväg som redan finns i produktion men som vilar tills inställningen slår på den, för alla eller för en utvald grupp användare. Pete Hodgson beskrev tekniken utförligt på Martin Fowlers webbplats, i avsnitt från januari 2016 och med en revidering i oktober 2017. Han kallar dem feature toggles och nämner feature flags, feature bits och feature flippers som andra namn på samma sak.

Hodgson delar upp en flagga i beståndsdelar. Toggle point är den plats i koden som frågar om den nya vägen är påslagen. Toggle router svarar på frågan utifrån flaggornas konfiguration och, när svaret beror på vem som frågar, utifrån en toggle context: oftast den aktuella användaren, eller en särskild cookie eller HTTP-header som står i användarens ställe. Han varnar för att koppla ihop de två för hårt. Koden vid toggle point bör ställa en snäv fråga, till exempel om den här orderbekräftelsen ska ha den nya länken för avbeställning, medan regeln som avgör det bor på ett ställe bakom routern, så att den kan ändras utan att man rör varje ställe i koden där frågan ställs.

Flaggor skiljer sig åt i hur länge de lever och hur ofta deras svar ändras, och Hodgson delar in dem i fyra slag. Release toggles låter ofärdig, otestad kod gå ut i produktion som latent kod, alltså kod som ännu inte körs; enligt Hodgson bör de i regel inte bli kvar mycket längre än en vecka eller två. Experiment toggles delar in användarna i grupper för ett A/B-test, där varje användare hamnar i samma grupp varje gång. Ops toggles låter dem som sköter driften ändra hur systemet beter sig i produktion. Många av dem är kortlivade, men några få långlivade nödbrytare (kill switches) låter driften skruva ner funktioner som systemet klarar sig utan när belastningen är ovanligt hög, till exempel en panel med rekommendationer som är dyr att generera; Hodgson jämför dem med en

manuellt styrd circuit breaker. Permissioning toggles avgör vilka funktioner en viss användare ser, till exempel premiumfunktioner för betalande kunder eller alfafunktioner för företagets egna anställda, och de kan bli kvar i åratal.

En funktionsflagga är inte en canary release, även om de två ofta används tillsammans. En canary kör två versioner av en tjänst sida vid sida och skickar en andel av trafiken till den nya; en flagga levererar en enda version som innehåller båda vägarna och väljer mellan dem vid varje anrop. En långlivad gren i versionshanteringen håller också nytt arbete åtskilt, men då når koden produktion först när grenen slås ihop, och då allt på en gång.

Använd begreppet i ett beslut

Använd ”Vad är en funktionsflagga?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Ett team som tar en idé från en skiss till något dina kunder kan logga in i, i stället för att lämna den vidare mellan tre leverantörer.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Ledde GIS-kartappen till 10 gånger högre intäkter — 10x intäktstillväxt, med kartprodukten i spetsen

Ledde GIS-kartappen till 10 gånger högre intäkter Hör av er →

Varför det spelar roll

Huvudskälet till att använda flaggor är att skilja driftsättning från lansering. Hodgson beskriver release toggles som det vanligaste sättet att omsätta en princip från Continuous Delivery i praktiken: lanseringen av en funktion är ett eget steg, skilt från driftsättningen av koden. Kod kan gå ut i produktion varje dag från en gemensam huvudgren, medan en funktion som tar en månad att bygga hålls dold tills den är klar, och det är det som gör trunkbaserad utveckling (trunk-based development) genomförbar.

Eftersom en flagga kan ge olika svar till olika användare, kan en lansering breddas stegvis. Hodgson beskriver hur nya funktioner först slås på för interna användare, något han kallar en Champagne Brunch, eftersom företaget får dricka sin egen champagne först. Därefter kommer en canary-grupp: i hans exempel slumpvis utvalda 1 procent av användarna, där urvalet görs med modulo av användar-id:t så att samma personer stannar kvar i gruppen, medan de övriga 99 procenten behåller det gamla beteendet. Rapporterar den gruppen problem, slås flaggan av, och ingen behöver rulla tillbaka en driftsättning.

Allt detta har ett pris. Varje flagga lägger till en förgrening i koden och en kombination som måste testas, och konfigurationen bakom den blir en del av produktionen som kan ändras utan ett nytt bygge. Hodgson råder till att hålla flaggornas konfiguration i versionshanteringen och ändra den genom nya driftsättningar där flaggan tillåter det, så att konfigurationen går genom samma pipeline som koden och ändringarna granskas och registreras. Konfiguration som ändras medan systemet kör gör nytta där en flagga måste kunna ändras inom några minuter, som ops toggles.

Så ser det ut i praktiken

flagd är en motor med öppen källkod för utvärdering av flaggor, och den följer OpenFeature-specifikationen. Den har inget användargränssnitt och lagrar ingenting själv; den läser flaggdefinitioner från en JSON-fil eller en annan källa och svarar på anrop, antingen som sidecar, inne i applikationens process eller som en central tjänst. En flagga som skickar ungefär var tjugonde användare till en ny kassa ser ut så här:

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

Varje flagga har ett tillstånd och en uppsättning varianter som alla måste vara av samma typ; standardvarianten är valfri. Regeln `fractional` fördelar användarna i hinkar efter relativ vikt, så 5 och 95 skickar ungefär var tjugonde användare till `on`. Som standard räknas hinken fram ur flaggnyckeln och `targetingKey` i utvärderingskontexten, vilket betyder att samma användare alltid får samma svar och att ett nytt namn på flaggan blandar om alla. `metadata` rymmer det som teamet vill knyta till flaggan, här en ägare och datumet då flaggan ska vara borta. Sätts `state` till `DISABLED`, svarar flagd utan något värde alls, och koden som frågar faller tillbaka på sitt eget standardvärde.

Kör flagd mot filen och be om ett svar över 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 innehåller värdet, varianten och skälet `TARGETING_MATCH`. I applikationen går anropet genom en OpenFeature-klient med flaggnyckeln, standardvärdet `false` och utvärderingskontexten, och toggle point ser inget annat än sant eller falskt. Att höja andelen från 5 till 20 procent är en ändring i flaggfilen, inte i applikationens bygge.

Arbetsytorna för rekrytering och sjöfolks karriärer, från 2025, visar en release-flagga i den dokumenterade ögonblicksbilden av implementeringen: den beskriver rekryteringssidan som i drift och karriärarbetsytan som liggande bakom en funktionsflagga och inte fullt lanserad. Det är en historisk ögonblicksbild, inte ett påstående om det aktuella driftstillståndet. Operatörernas backoffice från samma år är platsen där sådana flaggor ändras. Operatörerna slår om flaggor och slår på en funktion för bara en del av användarna från ett gränssnitt, i stället för att en ingenjör ändrar i produktionsdatabasen för hand, och varje ändring auktoriseras och registreras som vilken annan åtgärd som helst där.

Var det går fel

Flaggor som aldrig försvinner. En release-flagga som har gjort sitt är ren kostnad: en förgrening som ingen behöver, en väg som ingen testar och en inställning som någon kan råka slå om. Hodgson föreslår att man behandlar flaggor som ett lager som det kostar att hålla, och att man håller lagret litet.

Bland hans åtgärder finns en uppgift om flaggans borttagning som läggs i backloggen redan när en release-flagga införs, ett utgångsdatum, en tidsinställd bomb som får ett test att misslyckas eller hindrar applikationen från att starta när en flagga har levt förbi sitt datum, och ett fast tak för antalet flaggor, så att en ny bara kan läggas till när en annan tas bort. Att ta bort en flagga betyder att radera den förlorande kodvägen och inte bara kontrollen.

En gammal flagga med ny betydelse. Den 1 augusti 2012 skickade Knight Capitals automatiska orderrouter, SMARS, miljontals order ut på marknaden under 45 minuter, och företaget förlorade mer än 460 miljoner dollar. Beslutet från den amerikanska tillsynsmyndigheten SEC beskriver hur det gick till. Ny kod för New York-börsens Retail Liquidity Program återanvände en flagga som följde med inkommande order och som en gång hade slagit på Power Peg, en funktion som Knight hade slutat använda 2003 men aldrig tagit bort. En tekniker kopierade inte den nya koden till en av de åtta servrarna, så på den servern väckte den återanvända flaggan den gamla funktionen. När personalen sedan tog bort den nya koden från de sju andra, aktiverade flaggan Power Peg även där. Ge aldrig en pensionerad flagga en ny betydelse, radera död kod tillsammans med flaggan som skyddade den och kontrollera att alla servrar kör samma bygge innan en flagga slås på.

Kombinationer som ingen har testat. Två flaggor ger fyra kombinationer och tio ger över tusen; Hodgson kallar det en kombinatorisk explosion. Att testa alla är sällan möjligt, så testa de kombinationer som kan nå produktion: den aktuella produktionskonfigurationen med de flaggor som snart ska lanseras påslagna, reservläget med samma flaggor avslagna och, som många team också gör, allt påslaget. En fast konvention hjälper: av betyder alltid det befintliga beteendet och på betyder det nya.

Ett standardvärde som ingen valde. OpenFeature-specifikationen säger att utvärdering av en flagga inte får kasta undantag. När något går fel, returnerar anropet det standardvärde som anroparen skickade med, den detaljerade formen av resultatet innehåller en felkod, och klienten bör inte heller skriva till loggen, eftersom en applikation som driftsattes innan flaggan var definierad annars skulle fylla loggen med `FLAG_NOT_FOUND`. Det håller applikationen igång, men det betyder också att ett stavfel i en flaggnyckel i tysthet blir standardvärdet för alla användare. Skicka med det gamla, säkra beteendet som standardvärde och koppla ett tillägg till utvärderingen som räknar fel och slår larm när de dyker upp; det är också i ett sådant tillägg som specifikationen placerar loggningen.

Överstyrningar som vem som helst kan sätta. Det är bekvämt att låta testare tvinga på en flagga med en särskild cookie, en parameter i URL:en eller en HTTP-header. Hodgson påpekar att nyfikna eller illasinnade användare kan sätta samma överstyrning och ändra en flaggas tillstånd för egen del. Godta överstyrningar bara från intern trafik, eller signera dem kryptografiskt; det minskar risken, men som Hodgson konstaterar tillför det också komplexitet och en ny attackyta.

Läs mer

- Feature Toggles (aka Feature Flags) — Pete Hodgsons artikel på Martin Fowlers webbplats: delarna i en flagga, de fyra kategorierna, testning och hur man håller lagret av flaggor litet.
- Flag Evaluation API — OpenFeature-specifikationen för utvärdering av en flagga, bland annat standardvärden, felkoder och varför utvärderingen aldrig kastar undantag.
- flagd: Flag Definitions — JSON-formatet som flagd läser: tillstånd, varianter, standardvariant, målgruppsstyrning och metadata.
- flagd: Fractional Operation — deterministiska procentuella fördelningar och vad som avgör vilken hink en användare hamnar i.
- In the Matter of Knight Capital Americas LLC — SEC:s beslut från 2013 om handelsförlusten i augusti 2012, bland annat den återanvända flaggan.

Ordlista

Välj utifrån era faktiska förutsättningar

Beskriv arbetsflödet, befintliga gränssnitt och användarna som behöver förändringen. Vi kan definiera en testbar leverans och en överlämning som gör att teamet kan underhålla lösningen.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är en funktionsflagga?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

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

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/feature-flag/