Engineer

Blue-green deployment

◐ å flytte trafikken mellom to like produksjonsmiljøer

Hva det er

Blue-green deployment er en måte å gi ut programvare på med to produksjonsmiljøer som er bygd så like som mulig. Det ene, her kalt blue, betjener alle forespørsler. Den nye versjonen installeres i det andre, green, og testes der mens ingen kunder når fram til det. Når den er klar, endres det som står foran de to miljøene, slik at alle nye forespørsler går til green. Blue-miljøet kjører videre urørt, og oppfører green seg dårlig, setter den samme endringen gjort baklengs blue i drift igjen. På norsk brukes som regel det engelske navnet, ofte bare i kortformen blue-green.

Fargene betyr ingenting, og det er hele poenget. Navn som «gammel» og «ny» eller «primær» og «reserve» skifter betydning etter hver utgivelse, mens to farger bare bytter på: etter neste skifte er green i drift, og blue er det ledige miljøet. Martin Fowler beskrev teknikken på sin bliki i 2010 og tilskriver navnet Daniel Terhorst-North og Jez Humble, selv om han innrømmer at han husker opphavet noe uklart. Verktøyene staver det hver på sin måte: Argo Rollouts, en Kubernetes-kontroller som legger til utrullingsstrategier av dette slaget, kaller det `blueGreen` i konfigurasjonen sin.

Skiftet kan være et hvilket som helst enkeltpunkt som avgjør hvor forespørslene havner. I Fowlers versjon er det en ruter. Det kan like gjerne være målet til en lastbalanserer, upstream i en reverse proxy eller, i Kubernetes, en Service der label-selektoren peker ut hvilke pods som får trafikken. De to miljøene trenger heller ikke være to datasentre: Fowler nevner separat maskinvare, separate virtuelle maskiner eller to soner på samme system med hver sine IP-adresser som like gode løsninger.

To nabostrategier gjør forskjellen tydelig. En rullerende utrulling bytter ut instanser noen få om gangen, så gammel og ny versjon svarer side om side til den siste gamle er borte, og en tilbakerulling er enda en rullerende utrulling i motsatt retning. En canary release sender først en liten andel av trafikken til den nye versjonen og utvider den steg for steg. Blue-green flytter alt på én gang, men beholder en komplett, kjørende kopi av forrige versjon, så veien tilbake er ett steg, like stort som veien fram.

Bruk begrepet i en beslutning

Bruk «Hva er blue-green deployment?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Testet, utgitt og mulig å reversere, slik at endringer når kundene en tirsdag ettermiddag i stedet for sent en fredagskveld.

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. Automatiserte GIS-SaaS-utrulling i GitHub Actions — produksjonssetting, data og rapportering, alt uten manuelle grep

Automatiserte GIS-SaaS-utrulling i GitHub Actions Ta kontakt →

Hvorfor det er viktig

En utgivelse blir en kort hendelse som kan angres, i stedet for en lang enveistur. Det som tar tid, som å kopiere filer, starte prosesser og varme opp cacher, skjer i green mens blue fortsatt betjener kundene, så ingen venter på det. Selve skiftet går raskt, og det gjør angringen også: ingenting må bygges eller installeres på nytt, for forrige versjon sluttet aldri å kjøre. Det endrer hvordan et team har det med å gi ut en helt vanlig ettermiddag, og hvor raskt det kan handle når noe ser feil ut.

Green-miljøet kan dessuten prøves ut under forhold som ligger svært nær produksjon før noen kunde ser det: samme nettverk, samme konfigurasjon, samme database. En røyktest mot green beviser langt mer enn den samme testen i et stagingmiljø som avviker på et dusin små punkter. Argo Rollouts formaliserer dette med en preview-Service som sender trafikk til den nye versjonen før promoteringen, slik at den kan testes uten å betjene produksjonstrafikk.

Det ledige miljøet har en jobb til. Når en utgivelse har satt seg, blir siden som nettopp ble stille, stedet der neste utgivelse forberedes, og fram til da er den en varm reserve. Fowler påpeker at den samme mekanismen lar en organisasjon prøve ut planen sin for katastrofegjenoppretting ved hver eneste utgivelse, fordi hvert skifte er en failover som noen følger med på.

Prisen er lett å se. To produksjonsmiljøer krever kapasitet til to, i hvert fall mens en utgivelse pågår. Og selv om applikasjonslaget er lett å kopiere, er dataene det ikke: blue og green deler nesten alltid én database, så hver utgivelse må kunne leses av begge versjonene. Et team som ikke kan love det, får et raskt skifte fram og et skifte tilbake som ikke virker.

Slik ser det ut i praksis

I Kubernetes er den minste fungerende versjonen to Deployments og én Service. Deploymentene `web-blue` og `web-green` kjører den samme applikasjonen med labelene `app: web` og `slot: blue` eller `slot: green` . Servicen som kundene når, velger ett slot:

```
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  selector:
    app: web
    slot: blue
  ports:
    - port: 80
      targetPort: 8080
```

En utgivelse legger det nye imaget på den ledige siden, venter på det, tester det gjennom en ekstra Service som bare velger `slot: green` , og flytter så den Servicen som er i drift:

```
kubectl set image deployment/web-green web=registry.example.com/web:1.5.0
kubectl rollout status deployment/web-green --timeout=5m
# røyktester mot preview-Servicen for web-green her
kubectl patch service web -p '{"spec":{"selector":{"slot":"green"}}}'
# veien tilbake er samme kommando med den andre fargen
kubectl patch service web -p '{"spec":{"selector":{"slot":"blue"}}}'
```

`kubectl patch` bruker som standard en strategic merge patch, så bare nøkkelen `slot` i selektoren endres, mens `app: web` blir stående. Kontrolleren for Servicen leter hele tiden etter pods som matcher selektoren, og oppdaterer endepunktene når settet endrer seg, slik at nye forbindelser havner i green mens blue står klar med det forrige imaget.

Argo Rollouts automatiserer den samme dansen med ett `Rollout`-objekt: en `activeService` som den peker mot de nye podene ved promotering, og en valgfri `previewService` som når dem før. Med `autoPromotionEnabled: false` venter utrullingen før skiftet til noen kjører `kubectl argo rollouts promote web`.

Tjenesten DevOps og CI/CD-automatisering lover endringer som er testet, utgitt og mulige å reversere, og blue-green deployment er én måte å gjøre det siste billig på. Teknikken hviler på to vaner som produksjonssettingen av en plattform til Azure viser i bruk, uten et ekstra produksjonsslot: Bicep-malene holder målinfrastrukturen identisk fra miljø til miljø, og den samme bygde artefakten promoteres fra utvikling via staging til produksjon og plukker opp de riktige innstillingene der den havner. Den ruten fører en utgivelse gjennom miljøene ett etter ett. Blue-green setter en tvilling ved enden av den, og de samme to vanene, lik infrastruktur og én uendret artefakt, er det som lar et team stole på at tvillingen virkelig er lik.

Hvor det går galt

En skjemaendring gjør veien tilbake til en blindvei. Green starter, kjører en migrering som gir en kolonne nytt navn, og betjener trafikk uten problemer. Ti minutter senere sender en feil trafikken tilbake til blue, som fortsatt spør etter det gamle kolonnenavnet og feiler på hver eneste forespørsel. Tilbakerullingen var planlagt for applikasjonen, ikke for dataene. Fowlers råd fra 2015 er å skille de to: send først ut en databaseendring som begge versjonene kan arbeide med, deretter applikasjonen, og fjern det bare den gamle versjonen trengte når den nye har satt seg.

Tilstand skrevet i green blir strandet ved et skifte tilbake. Alt green holder for seg selv, for eksempel filer lastet opp til den lokale disken, sesjoner i minnet eller en egen cache med ventende jobber, finnes ikke i blue når trafikken vender tilbake. Brukerne blir logget ut eller savner det de nettopp lagret. Hold delt tilstand i lagre som begge fargene bruker. Må green ha egne data, foreslår Fowler å sende transaksjonene til begge miljøene, eller å starte green skrivebeskyttet og først åpne for skriving når det har vist seg å holde.

Veien tilbake skaleres bort før feilen viser seg. Argo Rollouts skalerer det gamle ReplicaSet ned etter skiftet, og `scaleDownDelaySeconds` er 30 som standard. En feil som først dukker opp etter et kvarter, en nattlig jobb eller en langsom

minnelekkasje, finner ingen blue å vende tilbake til. Sett forsinkelsen slik at den dekker tiden det pleier å ta før problemer viser seg, og godta at prisen er kapasiteten:

```
strategy:
  blueGreen:
    activeService: web
    previewService: web-preview
    autoPromotionEnabled: false
    scaleDownDelaySeconds: 3600
```

Promoteringen følger beredskap, ikke korrekthet. `autoPromotionEnabled` er true som standard, så Argo Rollouts skifter så snart de nye podene er klare. En readiness probe forteller at en prosess svarer, ikke at betalingsflyten virker. Hold pause før promoteringen som i blokken over og test gjennom preview-Servicen, eller legg til en `prePromotionAnalysis`, som avbryter utrullingen hvis den feiler, og en `postPromotionAnalysis`, som skifter tilbake til forrige versjon hvis den feiler etter skiftet.

Den ledige siden kommer i utakt. Mellom to utgivelser legger noen inn en hastende konfigurasjonsendring bare i miljøet som er i drift. Neste utgivelse lander på den ledige siden, som fortsatt har den gamle innstillingen, og skiftet som skulle endre én ting, endrer to. Bygg begge slotene fra de samme malene og den samme konfigurasjonskilden, så en endring i det ene er en endring i begge, og bruk den ledige siden som staging for neste utgivelse, slik Fowler beskriver det.

Les mer

- Blue Green Deployment — Martin Fowlers beskrivelse fra 2010, med tillegget hans fra 2015 om databaseendringer.
- Argo Rollouts: BlueGreen — aktiv og preview-Service, automatisk promotering, analyse før og etter skiftet og forsinkelsen på nedskaleringen.
- Kubernetes Service — hvordan en selektor peker ut hvilke pods som får trafikken til en Service.
- kubectl patch — strategic merge, merge og JSON-patcher, kommandoen som snur selektoren i eksempelet over.

Ordliste

Velg ut fra deres faktiske rammer

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er blue-green deployment?» 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/blue-green-deployment/