Engineer

Blue-green deployment

◐ at skifte trafikken mellem to ens produktionsmiljøer

Hvad det er

Blue-green deployment er en måde at udgive software på med to produktionsmiljøer, der er bygget så ens som muligt. Det ene, her kaldet blue, betjener alle forespørgsler. Den nye version installeres i det andet, green, og testes dér, mens ingen kunder rammer det. Når den er klar, ændres det, der står foran de to miljøer, så alle nye forespørgsler går til green. Blue-miljøet kører videre uberørt, og opfører green sig dårligt, sætter den samme ændring foretaget baglæns blue i drift igen. På dansk skrives det oftest på engelsk, men blue-green-udrulning ses også.

Farverne betyder ingenting, og det er hele pointen. Navne som »gammel« og »ny« eller »primær« og »standby« skifter betydning efter hver release, mens to farver blot tager tur: efter næste skift er green live, og blue er det ledige miljø. Martin Fowler beskrev teknikken på sin bliki i 2010 og tilskriver navnet Daniel Terhorst-North og Jez Humble, om end han selv kalder ophavet lidt tåget. Værktøjerne staver det hver på sin måde: Argo Rollouts, en Kubernetes-controller, der tilføjer udrulningsstrategier af den slags, kalder det `blueGreen` i sin konfiguration.

Skiftet kan være et hvilket som helst sted, der afgør, hvor forespørgslerne ender. I Fowlers udgave er det en router. Det kan lige så godt være målet på en load balancer, upstream i en reverse proxy eller i Kubernetes en Service, hvis label-selector udpeger de pods, der får trafikken. De to miljøer behøver heller ikke være to datacentre: Fowler nævner separat hardware, separate virtuelle maskiner eller to zoner på samme system med hver sine IP-adresser som lige gode løsninger.

To nabostrategier gør forskellen tydelig. En rullende opdatering skifter instanser ud nogle få ad gangen, så gammel og ny version svarer side om side, indtil den sidste gamle er væk, og en tilbagerulning er endnu en rullende opdatering i modsat retning. En canary release sender først en lille del af trafikken til den nye version og udvider skridt for skridt. Blue-green flytter alt på én gang, men beholder en komplet, kørende kopi af den forrige version, så vejen tilbage er ét skridt, lige så stort som vejen frem.

Brug begrebet i en beslutning

Brug »Hvad er blue-green deployment?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Testet, udgivet og til at rulle tilbage, så ændringer når kunderne en tirsdag eftermiddag i stedet for sent en fredag aften.

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 automatiseret GIS-SaaS-udrulning med CI/CD — udrulning, data og rapportering, alt sammen uden manuelt arbejde

Har automatiseret GIS-SaaS-udrulning med CI/CD Kontakt os →

Hvorfor det betyder noget

En release bliver en kort begivenhed, der kan fortrydes, i stedet for en lang envejstur. De langsomme dele, at kopiere filer, starte processer og varme caches op, sker i green, mens blue stadig betjener kunderne, så ingen venter på dem. Selve skiftet går hurtigt, og det gør fortrydelsen også: intet skal bygges eller installeres igen, for den forrige version holdt aldrig op med at køre. Det ændrer, hvordan et team har det med at udgive en helt almindelig eftermiddag, og hvor hurtigt det kan handle, når noget ser forkert ud.

Green-miljøet kan desuden afprøves under forhold, der ligger meget tæt på produktion, før nogen kunde ser det: samme netværk, samme konfiguration, samme database. En røgtest mod green beviser langt mere end den samme test i et stagingmiljø, der afviger på et dusin små punkter. Argo Rollouts formaliserer det med en preview-Service, der sender trafik til den nye version før promoveringen, så den kan testes uden at betjene produktionstrafik.

Det ledige miljø har et job mere. Når en release har lagt sig, bliver den side, der netop gik i ro, stedet, hvor den næste release forberedes, og indtil da er den et varmt standby-miljø. Fowler påpeger, at samme mekanisme lader en organisation afprøve sin plan for disaster recovery ved hver eneste release, fordi hvert skift er en failover, som nogen holder øje med.

Prisen er til at få øje på. To produktionsmiljøer kræver kapacitet til to, i hvert fald mens en release står på. Og selv om applikationslaget er let at kopiere, er data det ikke: blue og green deler næsten altid én database, så hver release skal kunne læses af begge versioner. Et team, der ikke kan love det, får et hurtigt skift frem og et skift tilbage, der ikke virker.

Sådan ser det ud i praksis

På Kubernetes er den mindste fungerende udgave to Deployments og én Service. Deploymentene `web-blue` og `web-green` kører den samme applikation med labels `app: web` og `slot: blue` eller `slot: green`. Den Service, kunderne rammer, vælger ét slot:

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

En release lægger det nye image på den ledige side, venter på det, tester det gennem en ekstra Service, der kun vælger `slot: green`, og flytter derefter den live Service:

```
kubectl set image deployment/web-green web=registry.example.com/web:1.5.0
kubectl rollout status deployment/web-green --timeout=5m
# røgtest mod preview-Servicen for web-green her
kubectl patch service web -p '{"spec":{"selector":{"slot":"green"}}}'
# vejen tilbage er samme kommando med den anden farve
kubectl patch service web -p '{"spec":{"selector":{"slot":"blue"}}}'
```

`kubectl patch` bruger som standard en strategic merge patch, så kun nøglen `slot` i selectoren ændres, mens `app: web` bliver stående. Servicens controller leder hele tiden efter pods, der matcher selectoren, og opdaterer Servicens endpoints, når mængden ændrer sig, så nye forbindelser lander i green, mens blue står klar med det forrige image.

Argo Rollouts automatiserer den samme dans med ét `Rollout`-objekt: en `activeService`, som den peger på de nye pods ved promoveringen, og en valgfri `previewService`, der når dem før. Med `autoPromotionEnabled: false` holder udrulningen pause før skiftet, indtil nogen kører `kubectl argo rollouts promote web`.

Ydelsen DevOps og CI/CD-automatisering lover ændringer, der er testet, udgivet og til at rulle tilbage, og blue-green deployment er én måde at gøre det sidste billigt på. Teknikken hviler på to vaner, som udrulningerne af en platform til Azure viser i brug, uden noget andet produktionsslot: Bicep-templates holder målinfrastrukturen ens fra det ene miljø til det næste, og det samme byggede artefakt promoveres fra udvikling over staging til produktion og henter de rette indstillinger der, hvor det lander. Den rute fører en release gennem miljøer på række. Blue-green sætter en tvilling for enden af den, og de samme to vaner, ens infrastruktur og ét uændret artefakt, er det, der lader et team stole på, at tvillingen virkelig er én.

Hvor det går galt

En skemaændring gør vejen tilbage til en blindgyde. Green starter, kører en migrering, der omdøber en kolonne, og betjener trafik uden problemer. Ti minutter senere sender en fejl trafikken tilbage til blue, som stadig spørger efter det gamle kolonnenavn og fejler på hver eneste forespørgsel. Tilbagerulningen var planlagt for applikationen, ikke for data. Fowlers råd fra 2015 er at skille de to ad: send først en databaseændring ud, som begge versioner kan arbejde med, så applikationen, og fjern det, kun den gamle version brugte, når den nye har lagt sig.

Tilstand skrevet i green strander ved et skift tilbage. Alt det, green holder for sig selv, for eksempel filer uploadet til den lokale disk, sessioner i hukommelsen eller en egen cache af ventende job, findes ikke i blue, når trafikken vender tilbage. Brugerne bliver logget ud eller mangler det, de lige har gemt. Hold delt tilstand i lagre, som begge farver bruger. Skal green have sine egne data, foreslår Fowler at sende transaktionerne til begge miljøer eller at starte green skrivebeskyttet og først åbne for skrivning, når det har vist sig at holde.

Vejen tilbage skaleres væk, før fejlen viser sig. Argo Rollouts skalerer det gamle ReplicaSet ned efter skiftet, og `scaleDownDelaySeconds` er som standard 30. En fejl, der først dukker op efter et kvarter, et natligt job eller en langsom

lækage, finder ingen blue at vende tilbage til. Sæt forsinkelsen, så den dækker den tid, det plejer at tage, før problemer viser sig, og accepter, at prisen er kapaciteten:

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

Promoveringen følger parathed, ikke korrekthed. `autoPromotionEnabled` er som standard sand, så Argo Rollouts skifter, så snart de nye pods er klar. En readiness probe fortæller, at en proces svarer, ikke at betalingsflowet virker. Hold pause før promoveringen som i blokken ovenfor og test gennem preview-Servicen, eller tilføj en `prePromotionAnalysis`, der afbryder udrulningen, hvis den fejler, og en `postPromotionAnalysis`, der skifter tilbage til den forrige version, hvis den fejler efter skiftet.

Den ledige side glider væk. Mellem to releases lægger nogen en hastende konfigurationsændring ind i det live miljø alene. Den næste release lander på den ledige side, som stadig har den gamle indstilling, og skiftet, der skulle ændre én ting, ændrer to. Byg begge slots ud fra de samme templates og den samme konfigurationskilde, så en ændring af det ene er en ændring af begge, og brug den ledige side som staging for næste release, sådan som Fowler beskriver det.

Læs mere

- Blue Green Deployment — Martin Fowlers beskrivelse fra 2010 med hans tilføjelse fra 2015 om databaseændringer.
- Argo Rollouts: BlueGreen — aktiv og preview-Service, automatisk promovering, analyse før og efter skiftet og forsinkelsen på nedskaleringen.
- Kubernetes Service — hvordan en selector udpeger de pods, der får en Services trafik.
- kubectl patch — strategic merge, merge og JSON-patches, kommandoen, der vender selectoren i eksemplet ovenfor.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er blue-green deployment?« 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/blue-green-deployment/