Engineer

DevOps

∞ praksis, der forener softwareudvikling og drift

Hvad det er

DevOps er en måde at organisere softwarearbejde på, hvor de, der skriver et system, også står for, hvordan det bygges, udgives og holdes kørende, og hvor trinene mellem at skrive kode og at køre den er automatiseret i stedet for at blive sendt fra ét team til det næste. Navnet er sat sammen af development og operations, altså udvikling og drift, de to grupper, det skulle bringe sammen.

NIST, det amerikanske National Institute of Standards and Technology, definerer det som praksis, der automatiserer processerne mellem udviklings- og driftsteams, så de kan bygge, teste og udgive software hurtigere og mere pålideligt. Definitionen er nyttig, fordi den peger på det, der betyder noget: processerne imellem. En virksomhed kan have fremragende udviklere og fremragende driftsfolk og alligevel tabe dage på det sted, hvor den ene overdrager en release til den anden.

I praksis viser DevOps sig som en håndfuld vaner, der støtter hinanden:

- **Fælles ejerskab af produktionen.** Den, der ændrer koden, ser, hvad ændringen gør, når den kører, gennem monitorering, logs og alarmer, og står til ansvar for den.
- **En automatiseret vej til release.** Build, test og udrulning kører som en pipeline, på samme måde hver gang, som regel oven på CI.
- **Infrastruktur skrevet som filer.** Servere, netværk og databaser beskrives i templates eller playbooks, der ligger ved siden af koden, så et miljø kan genopbygges i stedet for at blive husket.
- **Ét build, mange miljøer.** Det artefakt, der bestod sine tests i staging, er det samme, der når produktion; kun dets indstillinger er ændret.
- **Små, hyppige releases.** Hver enkelt indeholder få ændringer, så en fejl er let at finde og let at rulle tilbage.

DevOps er ikke et produkt og kun løst en jobtitel. Leverandører sælger værktøjer under navnet, og virksomheder ansætter »DevOps-ingeniører«, men hverken værktøjet eller titlen er selve tingen. En DevOps-ingeniør bygger og passer typisk de pipelines og den infrastruktur, der lader andre udviklere udgive på egen hånd, og det er en rolle inden for praksissen, ikke en erstatning for den.

Brug begrebet i en beslutning

Brug »Hvad er DevOps?« 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

Hver overdragelse mellem teams taber noget undervejs. Udvikleren ved, hvorfor en ændring blev lavet, og hvad den afhænger af; driftsfolkene ved, hvordan serverne ser ud, og hvad der gik i stykker i sidste måned. Når en release krydser fra den ene til den anden som en sag og en pakke, når begge halvdele af den viden frem ufuldstændige, og hullerne viser sig under udrulningen, hvor de er dyrest at finde.

The Twelve-Factor App, en metode til at bygge software-as-a-service-apps, navngiver tre af de huller præcist. Der er et hul i tid, når kode skrevet i dag først når produktion uger senere. Der er et hul i personer, når én gruppe skriver koden og en anden udruller den. Og der er et hul i værktøjer, når en bærbar kører én database og produktion en anden. DevOps er i høj grad et projekt for at lukke alle tre: udrul timer efter, at koden er skrevet, lad forfatterne selv udrulle, og hold alle miljøer så ens som muligt.

Release-størrelse er det andet argument. Googles bog om site reliability engineering siger det ligeud: hyppige releases giver færre ændringer mellem versionerne. En release med tre ændringer er let at ræsonnere om, når den går galt. En release med tre hundrede er en eftersøgning.

Det tredje er det, samme bog kalder toil: arbejde knyttet til at drive en tjeneste, som er manuelt, gentaget og muligt at automatisere, som er uden varig værdi, og som vokser i takt med tjenesten. Et team, der genstarter en proces i hånden

hver tirsdag, fornyer certifikater efter en kalenderpåmindelse og udvider diske, når en alarm går, bruger sin tid på arbejde, som et script ville gøre mere pålideligt. At automatisere det arbejde er den måde, et lille team holder trit med et voksende system på.

Sikkerhedsvejledningerne er landet på samme form. NIST's vejledning om DevSecOps for cloud-native applikationer tager udgangspunkt i en pipeline med trin for build, test, pakning, udrulning og drift, og regner infrastruktur, runtime-politikker og monitorering blandt de slags kode, pipelinen beskriver.

Sådan ser det ud i praksis

Siderne på dette site viser DevOps mindre som en titel end som et ansvarsområde: den samme ingeniør designer et system, får det i produktion og holder det der.

For et produkt til digitale kort omfattede grundlæggerens arbejde infrastruktur, databehandling og den kundevendte kortapplikation i to år. Arbejdet dækkede arkitektur, implementering og support på tværs af de lag: at designe, bygge og vedligeholde driftsmiljøet, databehandlingsworkflows og kortapplikationen. Det forbinder udvikling og drift i ét engagement. Beskrivelsen dokumenterer omfanget og varigheden af det ingeniørarbejde og den vedligeholdelse, ikke målinger af oppetid eller svartider.

På en databasepark med 10 PostgreSQL- og MS SQL Server-instanser på Ubuntu Linux VPS blev driftssiden automatiseret stykke for stykke: backups med `pg_dump` og SQL Server Agent-jobs, overvågning i Prometheus og Grafana, regelmæssig patching, UFW-firewalls, rollebaseret adgangskontrol og dokumenteret disaster recovery. Siden melder 99,9 % oppetid på tværs af alle ti servere og manuelt vedligeholdelsesarbejde reduceret med 50 % gennem automatisering, hvilket er toil-argumentet målt på én databasepark.

På en Azure-platform med udviklings-, staging- og produktionsmiljøer blev udrulningen ejet fra ende til anden ad en fast rute. Udrulningspipelinen bygger images, Bicep-templates giver hvert miljø den samme infrastruktur, og den konfiguration, der skifter mellem miljøerne, ligger adskilt fra secrets. Det lader det samme byggede artefakt blive promoveret gennem miljøerne i stedet for at blive bygget om for hvert, og siden melder releases uden ad hoc-manuelle deploys i blandingen.

Hvor det går galt

DevOps svigter stille, som regel ved at beholde ordforrådet og miste vanerne.

Titlen kommer, og muren bliver stående. Et driftsteam får et nyt navn, eller et »DevOps-team« ansættes til at sidde mellem udviklerne og produktionen. Releases krydser stadig en overdragelse; den har bare fået et nyt navn på den anden side. Testen er enkel: spørg, hvem der kan udgive en ændring, og hvem der bliver vækket, når den går i stykker. Er det forskellige personer med forskellig viden, er praksissen ikke på plads.

Udrulning er stadig en person med en tjekliste. Pipelinen bygger og tester, og så logger nogen ind på en server og kører de sidste fem trin efter hukommelsen, en smule forskelligt hver gang. Læg også de sidste trin i pipelinen, giv releases én rute gennem miljøerne, og behandl et manuelt push til produktion som en hændelse, der skal forklares.

Hvert miljø får sit eget build. At bygge om til staging og igen til produktion betyder, at det, der blev testet, ikke er det, der blev udgivet. En afhængighed, der blev løst anderledes ved andet build, dukker op som en fejl, som ingen kan genskabe. Byg én gang, mærk artefaktet med det commit, det kom fra, og promovér netop det artefakt, med indstillingerne hentet fra miljøet:

```
tag="$(git rev-parse --short HEAD)"
docker build -t "registry.example.com/app:$tag" .
docker push "registry.example.com/app:$tag"
# staging og produktion kører præcis dette tag;
# kun indstillingerne, det startes med, er forskellige
docker run --env-file staging.env "registry.example.com/app:$tag"
```

Staging glider væk fra produktion. En indstilling ændret i hånden på én server, en database opgraderet i produktion først, en biblioteksversion låst ét sted og flydende et andet: hver af dem gør staging en smule mindre brugbar som generalprøve. Beskriv infrastrukturen i templates, anvend de samme templates overalt, og kør de samme bagvedliggende tjenester i udvikling, som produktion kører.

Toil vokser med systemet. Manuelle opgaver, der tog ti minutter om ugen for én server, tager timer for tyve. Uden et bevidst budget til at automatisere dem ender teamet med at bruge dagene på at holde lyset tændt. Tæl det tilbagevendende manuelle arbejde, og sæt et loft over, hvor stor en andel af tiden det må tage.

Læs mere

- DevOps — NIST's definition med de publikationer, den er hentet fra.
- Build, release, run — hvorfor et build laves én gang og kombineres med konfiguration til en release.
- Dev/prod parity — hullerne i tid, personer og værktøjer mellem udvikling og produktion, og hvordan de lukkes.
- Config — alt, der varierer mellem deploys, holdt ude af koden og placeret i miljøet.
- Eliminating Toil — Googles definition af toil og argumentet for at sætte et loft over den.
- Release Engineering — hermetiske builds, selvbetjente releases og politikker, som release-værktøjerne håndhæver.



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 DevOps?« 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/devops/