Engineer

DevOps

∞ praksiser som forener programvareutvikling og drift

Hva det er

DevOps betegner en måte å organisere programvarearbeid på der de som skriver et system, også eier hvordan det bygges, gis ut og holdes i drift, og der stegene mellom ferdig kode og kjørende tjeneste er automatisert i stedet for å bli sendt fra ett team til et annet. Navnet er satt sammen av development og operations, utvikling og drift, de to gruppene det ble laget for å bringe sammen. På norsk har ordet ingen etablert oversettelse, og de fleste skriver det som på engelsk.

NIST, det amerikanske standardiseringsinstituttet, definerer DevOps som praksiser for å automatisere prosessene mellom utviklings- og driftsteam, slik at de kan bygge, teste og gi ut programvare raskere og mer pålitelig. Definisjonen er nyttig fordi den peker på den delen som betyr noe: prosessene imellom. Et selskap kan ha dyktige utviklere og dyktige driftsfolk og likevel tape dager i øyeblikket der den ene gruppen leverer en utgivelse over til den andre.

I praksis viser DevOps seg som en håndfull vaner som forsterker hverandre:

- **Felles eierskap til produksjon.** Den som endrer koden, ser hva endringen gjør når den kjører, gjennom overvåking, logger og et driftsvarsel når noe svikter, og står til ansvar for den.
- **En automatisert vei til utgivelse.** Bygging, testing og produksjonssetting kjører som en leveransepipeline, likt hver gang, som regel oppå CI.
- **Infrastruktur skrevet som filer.** Servere, nettverk og databaser beskrives i maler eller playbooks som ligger ved siden av koden, slik at et miljø kan bygges opp på nytt i stedet for å huskes.
- **Ett bygg, mange miljøer.** Artefakten som besto testene i staging, er den samme som når produksjon; bare innstillingene er endret.
- **Små, hyppige utgivelser.** Hver av dem bærer få endringer, så en feil er lett å finne og lett å rulle tilbake.

DevOps er ikke et produkt, og bare løselig en stillingstittel. Leverandører selger verktøy under navnet, og selskaper ansetter «DevOps-ingeniører», men ingen av delene er selve tingen. En DevOps-ingeniør bygger og passer som regel de leveransepipelinene og den infrastrukturen som lar andre utviklere levere på egen hånd. Det er en rolle inne i praksisen, ikke en erstatning for den.

Bruk begrepet i en beslutning

Bruk «Hva er DevOps?» 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

Hver overlevering mellom to team mister noe på veien. Utvikleren vet hvorfor en endring ble gjort, og hva den er avhengig av; driftsfolkene vet hvordan serverne ser ut, og hva som gikk i stykker forrige måned. Når en utgivelse krysser fra den ene til den andre som en sak og en pakke, kommer begge halvdelene av den kunnskapen ufullstendig fram. Hullene viser seg under produksjonssettingen, og det er det dyreste tidspunktet å oppdage dem på.

The Twelve-Factor App, en metodikk for å bygge programvare som leveres som tjeneste, navngir tre av disse hullene presist:

- **Et hull i tid:** kode skrevet i dag når produksjon uker senere.
- **Et hull i folk:** én gruppe skriver koden, og en annen setter den i produksjon.
- **Et hull i verktøy:** en bærbar maskin kjører én database, og produksjon kjører en annen.

DevOps er i stor grad et prosjekt for å lukke alle tre: sett i produksjon timer etter at koden er skrevet, la dem som skrev den, også sette den i produksjon, og hold alle miljøer så like som mulig.

Størrelsen på utgivelsene er det andre argumentet. Googles bok om site reliability engineering sier det rett ut: hyppige utgivelser betyr færre endringer mellom versjonene. En utgivelse med tre endringer i er lett å resonnere om når den går galt. En utgivelse med tre hundre er et søk.

Det tredje argumentet er det boka kaller toil, altså rutinearbeid. Den definerer det som arbeid knyttet til å drive en tjeneste som er manuelt, repeterende og mulig å automatisere, som ikke gir varig verdi, og som vokser i takt med tjenesten. Et team som starter en prosess på nytt for hånd hver tirsdag, fornyer sertifikater etter en påminnelse i kalenderen og utvider disker når et driftsvarsel går, bruker tiden sin på arbeid som et skript ville gjort mer pålitelig. Å automatisere det arbeidet er måten et lite team holder tritt med et voksende system på.

Sikkerhetsveiledningene har landet på den samme formen. NISTs veiledning om DevSecOps for skynative applikasjoner tar utgangspunkt i en leveransepipeline med trinn for bygging, testing, pakking, produksjonssetting og drift, og regner infrastruktur, kjøretidspolicy og overvåking blant de typene kode som leveransepipelinen beskriver.

Slik ser det ut i praksis

Sidene på dette nettstedet viser DevOps mindre som en tittel enn som et ansvarsområde: den samme ingeniøren utformer et system, får det i produksjon og holder det der.

For et kartprodukt spente grunnleggerens arbeid over infrastruktur, databehandling og kartapplikasjonen for kundene i to år. Arbeidet omfattet arkitektur, implementering og støtte på tvers av disse lagene: å utforme, bygge og vedlikeholde driftsmiljøet, arbeidsflytene for databehandling og kartapplikasjonen. Det knytter utvikling og drift sammen i ett oppdrag. Dokumentasjonen fastslår omfanget og varigheten av dette utviklings- og vedlikeholdsarbeidet, ikke målinger av oppetid eller responstid.

I en databasepark med 10 PostgreSQL- og MS SQL Server-instanser på VPS-er med Ubuntu Linux ble driftssiden automatisert bit for bit: sikkerhetskopier med `pg_dump` og SQL Server Agent-jobber, overvåking i Prometheus og Grafana, regelmessig patching, UFW-brannmurer, rollebasert tilgangskontroll og

dokumenterte prosedyrer for katastrofegjenoppretting. Siden melder 99,9 % oppetid på alle de ti serverne og 50 % mindre manuelt vedlikehold takket være automatisering. Det er argumentet om rutinearbeid, målt på én park.

På en Azure-plattform med utviklings-, staging- og produksjonsmiljø ble produksjonssettingen eid fra ende til annen langs en fast rute. Leveransepipelinen bygger imagene, Bicep-malene holder infrastrukturen identisk fra miljø til miljø, og hemmelighetene ligger for seg selv, skilt fra innstillingene som varierer fra miljø til miljø. Det lar den samme bygde artefakten promoteres gjennom miljøene i stedet for å bygges på nytt for hvert av dem, og siden melder om utgivelser uten ad hoc-produksjonssettinger for hånd innimellom.

Hvor det går galt

DevOps svikter stille, og som regel ved at ordene blir værende mens vanene forsvinner.

Tittelen kommer, og muren blir stående. Et driftsteam får nytt navn, eller et «DevOps-team» ansettes for å sitte mellom utviklerne og produksjon. Utgivelsene krysser fortsatt en overlevering; den har bare fått et nytt navn på den andre siden. Testen er enkel: spør hvem som kan levere en endring, og hvem som blir vekket når den går i stykker. Er det to forskjellige personer med forskjellig informasjon, har praksisen ikke kommet fram.

Produksjonssetting er fortsatt en person med en sjekkliste. Pipelinen bygger og tester, og så logger noen seg inn på en server og kjører de siste fem stegene etter hukommelsen, litt annerledes hver gang. Legg også de siste stegene inn i leveransepipelinen, gi utgivelsene én rute gjennom miljøene, og behandle en manuell push til produksjon som en driftshendelse som skal forklares.

Hvert miljø får sitt eget bygg. Å bygge på nytt for staging og så igjen for produksjon betyr at det som ble testet, ikke er det som ble levert. En avhengighet der en annen versjon ble valgt i det andre bygget, dukker opp som en feil ingen klarer å gjenskape. Bygg én gang, merk artefakten med commiten den kom fra, og promoter nettopp den artefakten, med innstillingene hentet fra miljøet:

```
tag="$(git rev-parse --short HEAD)"
docker build -t "registry.example.com/app:$tag" .
docker push "registry.example.com/app:$tag"
# staging og produksjon kjører den samme taggen;
# bare innstillingene de startes med, er forskjellige
docker run --env-file staging.env "registry.example.com/app:$tag"
```

Staging glir bort fra produksjon. En innstilling endret for hånd på én server, en database oppgradert i produksjon først, en bibliotekversjon låst ett sted og flytende et annet: hver av dem gjør staging litt mindre nyttig som generalprøve. Beskriv infrastrukturen i maler, bruk de samme malene overalt, og kjør de samme støttetjenestene i utvikling som produksjon kjører.

Rutinearbeidet vokser med systemet. Manuelle jobber som tok ti minutter i uka for én server, tar timer for tjue. Uten et bevisst budsjett for å automatisere dem ender teamet med å bruke dagene på å holde driften gående. Tell det gjentakende manuelle arbeidet, og sett et tak på hvor stor andel av tiden det får ta.

Les mer

- DevOps — NISTs definisjon, med spesialpublikasjonene den er hentet fra.
- Build, release, run — hvorfor et bygg lages én gang og kombineres med konfigurasjon til en utgivelse.
- Dev/prod parity — hullene i tid, folk og verktøy mellom utvikling og produksjon, og hvordan de lukkes.
- Config — å holde alt som varierer mellom produksjonssettinger, ute av koden og i miljøet.
- Eliminating Toil — Googles definisjon av toil, og argumentet for å sette et tak på det.
- Release Engineering — hermetiske bygg, selvbetjente utgivelser og regler som håndheves av utgivelsesverktøyene.

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 DevOps?» 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/devops/