Engineer

DevOps

∞ metoder som förenar mjukvaruutveckling och drift

Vad det är

DevOps är ett sätt att organisera arbetet med mjukvara där de som skriver ett system också äger hur det byggs, släpps och hålls igång, och där stegen mellan att skriva kod och att köra den är automatiserade i stället för att lämnas över från ett team till ett annat. Namnet är en sammandragning av development och operations, utveckling och drift, de två grupper som ordet myntades för att föra samman.

NIST, det amerikanska institutet för standarder och teknik, definierar DevOps som metoder för att automatisera processerna mellan utvecklingsteam och driftteam, så att teamen kan bygga, testa och släppa mjukvara snabbare och mer tillförlitligt. Definitionen är användbar eftersom den pekar ut den del som avgör: processerna däremellan. Ett företag kan ha utmärkta utvecklare och utmärkta driftstekniker och ändå förlora dagar i det ögonblick då den ena gruppen lämnar en release till den andra.

I praktiken märks DevOps som en handfull vanor som stöttar varandra:

- **Delat ägarskap av produktionen.** Den som ändrar koden ser vad ändringen gör när den väl körs, genom övervakning, loggar och larm, och svarar för den.
- **En automatiserad väg till release.** Bygge, test och driftsättning körs som en pipeline, på samma sätt varje gång, oftast ovanpå CI.
- **Infrastruktur skriven som filer.** Servrar, nätverk och databaser beskrivs i mallar eller playbooks som ligger bredvid koden, så att en miljö kan byggas upp på nytt i stället för att någon måste minnas hur den är uppbyggd.
- **Ett bygge, många miljöer.** Artefakten som klarade sina tester i staging är samma artefakt som når produktion, och bara dess inställningar har ändrats.
- **Små, täta releaser.** Var och en bär få ändringar, så ett fel är lätt att hitta och lätt att rulla tillbaka.

DevOps är ett arbetssätt snarare än en produkt, och en yrkestitel bara i lös mening. Leverantörer säljer verktyg under namnet och företag anställer ”DevOps-ingenjörer”, men varken verktygen eller titeln är själva saken. En DevOps-ingenjör bygger och sköter oftast de pipelines och den infrastruktur som låter andra utvecklare leverera på egen hand, vilket är en roll inom arbetssättet och ingen ersättning för det.

Använd begreppet i ett beslut

Använd ”Vad är DevOps?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Testade, släppta och möjliga att backa, så att ändringar når kunderna en tisdagseftermiddag i stället för sent en fredagskväll.

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. Automatiserade GIS-driftsättning i GitHub Actions — driftsättning, data och rapportering, helt utan manuellt arbete

Automatiserade GIS-driftsättning i GitHub Actions Hör av er →

Varför det spelar roll

Varje överlämning mellan team tappar något på vägen. Utvecklaren vet varför en ändring gjordes och vad den är beroende av; drifttekniken vet hur servrarna ser ut och vad som gick sönder förra månaden. När en release går från den ena till den andra som ett ärende och ett paket kommer båda halvorna av den kunskapen fram ofullständiga, och luckorna visar sig under driftsättningen, som är den dyraste tidpunkten att hitta dem.

The Twelve-Factor App, en metod för att bygga applikationer som levereras som tjänst, sätter exakta namn på tre av de luckorna. Det finns en lucka i tid, när kod som skrivs i dag når produktion flera veckor senare. Det finns en lucka i personer, när en grupp skriver koden och en annan driftsätter den. Och det finns en lucka i verktyg, när en bärbar dator kör en databas och produktionen kör en annan. DevOps är i stor utsträckning ett projekt för att stänga alla tre: driftsätt några timmar efter att koden skrevs, låt dem som skrev den driftsätta den, och håll varje miljö så lik de andra som möjligt.

Releasernas storlek är det andra argumentet. Boken från Google om site reliability engineering säger det rakt ut: täta releaser innebär färre ändringar mellan versionerna. En release med tre ändringar i är lätt att resonera kring när den går fel. En release med trehundra är ett sökarbete.

Det tredje är det som boken kallar toil, ungefär manuellt rutinslit. Den definierar toil som arbete knutet till att driva en tjänst som är manuellt, repetitivt och går att automatisera, som saknar bestående värde och som växer i takt med tjänsten. Ett team som startar om en process för hand varje tisdag, förnyar certifikat efter en påminnelse i kalendern och förstorar diskar när ett larm går ägnar sin tid åt arbete som ett skript skulle utföra mer tillförlitligt. Att automatisera det arbetet är hur ett litet team hinner med ett system som växer.

Vägledningen inom informationssäkerhet har landat i samma form. Vägledningen från NIST om DevSecOps för molnnativa applikationer utgår från en pipeline med steg för bygge, test, paketering, driftsättning och drift, och räknar infrastruktur, policyer för körning och övervakning till de slags kod som pipelinen beskriver.

Så ser det ut i praktiken

Sidorna på den här webbplatsen visar DevOps mindre som en titel än som ett ansvarsområde: samma ingenjör utformar ett system, får ut det i produktion och håller det kvar där.

För en kartprodukt omfattade grundarens arbete infrastruktur, databearbetning och den kundinriktade kartapplikationen under två år. Arbetet täckte arkitektur, implementering och support i de olika lagren: att utforma, bygga och underhålla driftsmiljön, arbetsflöden för databearbetning och kartapplikationen. Det förbinder utveckling och drift i ett uppdrag. Beskrivningen fastställer omfattningen och varaktigheten av det tekniska arbetet och underhållet, snarare än mätningar av upptid eller svarstider.

På en databasmiljö med 10 instanser av PostgreSQL och MS SQL Server på VPS:er med Ubuntu Linux automatiserades driften bit för bit: säkerhetskopior med `pg_dump` och jobb i SQL Server Agent, övervakning i Prometheus och Grafana, regelbunden patchning, brandväggar med UFW, rollbaserad åtkomstkontroll och dokumenterade rutiner för katastrofåterställning. Sidan redovisar 99,9 % upptid på alla tio servrar och 50 % mindre manuellt underhåll tack vare automatisering, vilket är argumentet om rutinslit mätt i en enda miljö.

På en plattform i Azure med miljöer för utveckling, staging och produktion ägdes driftsättningen från början till slut längs en fast väg. CI/CD-pipelinen bygger avbildningarna, mallar i Bicep håller infrastrukturen likadan från en miljö till nästa, och den konfiguration som skiljer sig mellan miljöerna hålls åtskild från

hemligheterna. Det låter samma byggda artefakt föras vidare genom miljöerna i stället för att byggas om för var och en, och sidan redovisar releaser helt fria från manuella ad hoc-driftsättningar.

Var det går fel

DevOps töms oftast på innehåll i det tysta: orden blir kvar medan vanorna försvinner.

Titeln kommer och muren står kvar. Ett driftteam byter namn, eller så anställs ett ”DevOps-team” som ska sitta mellan utvecklarna och produktionen. Releaserna passerar fortfarande en överlämning; den har bara fått ett nytt namn på andra sidan. Testet är enkelt: fråga vem som kan leverera en ändring och vem som blir väckt när den går sönder. Är det olika personer med olika information har arbetssättet ännu inte kommit på plats.

Driftsättningen är fortfarande en person med en checklista. Pipelinen bygger och testar, och sedan loggar någon in på en server och kör de fem sista stegen ur minnet, lite olika varje gång. Lägg även de sista stegen i pipelinen, ge releaserna en enda väg genom miljöerna och behandla en manuell push till produktion som en incident som ska förklaras.

Varje miljö får sitt eget bygge. Att bygga om för staging och sedan igen för produktion innebär att det som testades är en annan sak än det som levererades. Ett beroende där en annan version valdes vid det andra bygget dyker upp som ett fel som ingen kan återskapa. Bygg en gång, tagga artefakten med den commit den kom från och för den artefakten vidare, med inställningarna hämtade från miljön:

```
tag="$(git rev-parse --short HEAD)"
docker build -t "registry.example.com/app:$tag" .
docker push "registry.example.com/app:$tag"
# staging and production run this same tag;
# only the settings they start it with differ
docker run --env-file staging.env "registry.example.com/app:$tag"
```

Staging glider ifrån produktionen. En inställning som ändrats för hand på en server, en databas som uppgraderats i produktion först, en biblioteksversion som är låst på ett ställe och flytande på ett annat: var och en gör staging lite mindre

användbar som generalrepetition. Beskriv infrastrukturen i mallar, tillämpa samma mallar överallt och kör samma bakomliggande tjänster i utvecklingsmiljön som produktionen kör.

Rutinslitet växer med systemet. Manuella jobb som tog tio minuter i veckan för en server tar timmar för tjugo. Om ingen avsätter tid för att automatisera dem går teamets dagar åt till att hålla lamporna tända. Räkna det återkommande manuella arbetet och sätt ett tak för hur stor andel av tiden det får ta.

Läs mer

- DevOps — definitionen från NIST, med de specialpublikationer den bygger på.
- Build, release, run — varför ett bygge görs en gång och kombineras med konfiguration till en release.
- Dev/prod parity — luckorna i tid, personer och verktyg mellan utveckling och produktion, och hur de stängs.
- Config — att hålla allt som skiljer sig mellan driftsättningar borta från koden och i miljön.
- Eliminating Toil — definitionen av toil från Google och argumenten för att sätta ett tak för det.
- Release Engineering — hermetiska byggen, releaser i självbetjäning och policyer som releaseverktygen själva upprätthåller.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är DevOps?” 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/devops/