Engineer

Driftsättning

att flytta en ändring dit människor faktiskt använder den

Vad det är

Driftsättning är den följd av steg som tar en testad version av en applikation ut ur byggsystemet och sätter den i tjänst. De byggda filerna eller containeravbildningen placeras på de maskiner eller den plattform som ska köra dem, får konfigurationen för just den miljön och startas, och sedan flyttas verkliga anrop över till den nya versionen. Ordet står både för handlingen och för resultatet, så ett team talar om en driftsättning som fastnade i databassteget och om de driftsättningar som just nu körs i varje miljö. Det som driftsätts kan vara en webbplats, ett API, ett schemalagt jobb, en ändring i en databas eller den backend som en mobilapp pratar med. I tal säger många svenska utvecklare deploy och deploya; i skrift är driftsättning det vanliga ordet.

Att driftsätta och att släppa en release hålls ofta isär. Driftsättningen placerar den nya koden där den kan köras; releasen avgör vem som får se den. En funktion som döljs bakom en funktionsflagga kan driftsättas på måndagen och släppas till kunderna på torsdagen, utan att någon rör servrarna igen. Att hålla isär de två låter ett team leverera kod i små, täta steg, medan besluten om när något ska nå kunderna ligger kvar hos dem som äger produkten.

Många driftsättningar levererar i dag en containeravbildning. Open Container Initiative, som bildades inom Linux Foundation i juni 2015, publicerar tre specifikationer kring den: hur en avbildning är uppbyggd, hur en containerkörmiljö kör den när den har packats upp, och hur register lagrar och delar ut avbildningar. En avbildning identifieras av sin digest, en hash av innehållet, så samma digest betyder alltid samma byte. En tagg som `1.4.2` är en läsbar pekare till en avbildning, och den som pushar igen under samma tagg pekar om den till något nytt.

The Twelve-Factor App, en ofta citerad metod för att bygga webbtjänster, delar upp vägen från kod till körande tjänst i tre steg som hålls strikt åtskilda. Byggsteget gör om en commit till ett körbart paket. Releasesteget kombinerar paketet med konfigurationen för en viss miljö. Körsteget startar processer från en vald release. Varje release får en unik identifierare och ändras aldrig när den väl finns; varje ändring ger en ny release. Det är det som gör en återgång, att rulla tillbaka, till en fråga om att välja en tidigare release.

Hur bytet sker beror på vad tjänsten tål. En driftsättning av typen recreate stoppar den gamla versionen, startar sedan den nya och accepterar ett glapp däremellan. En rullande driftsättning byter ut instanser några i taget, så att båda versionerna betjänar trafik en stund; det är standardbeteendet för ett Deployment i Kubernetes. En blå-grön driftsättning kör den nya versionen bredvid den gamla och flyttar all trafik på en gång, med den gamla redo att ta tillbaka den. En canary skickar först en liten andel av trafiken till den nya versionen och breddar andelen så länge felfrekvensen och latensen håller sig friska.

Använd begreppet i ett beslut

Använd ”Vad är driftsättning?” 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

Driftsättningen är den punkt där en ändring börjar hjälpa någon, och också den punkt där ett misstag börjar nå användare. Team som driftsätter sällan samlar gärna många ändringar i varje release, så varje driftsättning bär större risk och, när den går sönder, fler kandidater till orsaken. Små, täta driftsättningar håller varje steg lätt att förstå och lätt att backa.

En driftsättning som görs för hand är precis så konsekvent som personen som gör den. Steg hoppas över, görs i en annan ordning eller görs lite annorlunda av en kollega som täcker upp under en semester. En CI/CD-pipeline utför samma steg varje gång och lämnar efter sig en uppgift om vilken commit, vilken artefakt

och vilken miljö det gällde, med en tidsstämpel. Den uppgiften är där granskningen efter en incident börjar, och ofta det snabbaste sättet att svara på vad som ändrades.

Den starkaste vanan är att bygga en gång och föra resultatet vidare. Artefakten som testades i staging ska byte för byte vara den som når produktion, och bara konfigurationen ska skilja sig; miljön levererar den konfigurationen när artefakten kommer fram. Registrera digesten vid bygget och driftsätt efter digest, så går löftet att kontrollera:

```
podman build -t registry.example.com/web:1.4.2 .
podman push --digestfile web.digest registry.example.com/web:1.4.2
kubectl -n staging set image deployment/web \
  web=registry.example.com/web@$(cat web.digest)
# after staging checks pass, the same digest goes to production
kubectl -n production set image deployment/web \
  web=registry.example.com/web@$(cat web.digest)
```

Att rulla tillbaka vilar på samma disciplin. När releaser är oföränderliga och har en identitet innebär en återgång att peka på den föregående, vilket tar sekunder. När varje miljö bygger om från källkoden innebär en återgång att bygga om en gammal commit och hoppas att samma beroendeversioner fortfarande väljs.

Så ser det ut i praktiken

Tre projekt på webbplatsen visar driftsättning som en väg som utformas en gång och sedan följs.

Driftsättningarna av en plattform till Azure ägdes från början till slut, och varje ändring följde samma fasta väg från utveckling via staging till produktion, utan att någon pushade direkt till en skarp miljö. CI/CD-pipelinen bygger avbildningarna och levererar dem, medan mallar i Bicep håller målinfrastrukturen likadan i varje miljö, så att varje steg vidare landar på matchande infrastruktur. Inställningar som skiljer sig mellan miljöerna hålls åtskilda från hemligheterna, vilket låter ett enda bygge färdas genom varje steg och plocka upp rätt inställningar där det hamnar, utan att byggas om på vägen. Sidan beskriver hur ändringar når varje miljö längs den fastställda vägen utan manuella ad hoc-driftsättningar, och hur releasen blev ett kontrollerat steg som går till på samma sätt varje gång.

Ett tidigare uppdrag visar vad den konsekvensen är värd i timmar. Automatiserade pipelines för bygge, test och driftsättning sparade ungefär 4 000 timmar genom att standardisera vägen från en ändring till produktion och ta bort manuella steg som varierade med vem som utförde dem. Sidan beskriver hur releaserna blev både snabbare och mer tillförlitliga.

På företagets egen plattform hålls driftsättning åtskild från provisionering. En containermiljö byggd på Podman och Quadlet låter root förbereda driftsättningsvägen en gång. Därefter driftsätter en oprivilegierad användare på nytt genom att köra ett enda fast skript via en snävt avgränsad regel som tillåter just det kommandot utan argument. Skriptet bygger avbildningen och startar om tjänsten, så en rutinmässig release kräver varken ett root-skal eller godtyckliga kommandon.

Var det går fel

De flesta problem med en driftsättning kommer av en skillnad mellan det som kontrollerades och det som faktiskt ändrades.

Produktionen kör något som staging aldrig såg. När varje miljö bygger sin egen kopia väljs nyare versioner inom beroendenas versionsintervall, en basavbildning flyttar sig mellan två byggen eller en byggflagga skiljer sig, och artefakten som klarade testerna är en annan än den som betjänar kunderna. Bygg en gång, registrera digesten, för vidare just den digesten, och håll allt som är specifikt för en miljö i konfigurationen snarare än i artefakten.

En tagg flyttar sig, och en återgång landar på fel version. En driftsättning som anger `latest`, eller som återanvänder en versionstagg efter ett nytt bygge, beror på vad taggen pekar på den dag den körs. Dokumentationen för Kubernetes avråder från `:latest` i produktion, eftersom det då blir svårare att avgöra vilken version som körs och svårare att rulla tillbaka på rätt sätt. Använd en versionstagg som aldrig återanvänds, och lås fast digesten där plattformen tillåter det.

Pipelinen rapporterar framgång innan den nya versionen fungerar. Många kommandon för driftsättning returnerar så snart plattformen har tagit emot begäran, medan de nya instanserna fortfarande startar och kanske kraschar en minut senare. Ett steg som väntar tills den nya versionen rapporterar att den är

redo, och sedan skickar några verkliga anrop och granskar svaren, gör ett tyst fel till ett rött bygge medan den föregående releasen fortfarande finns nära till hands.

Ny kod och den gamla tabellstrukturen möts mitt i en utrullning. Under en rullande driftsättning eller en canary betjänar gamla och nya versioner trafik sida vid sida mot en och samma databas. En databasmigrering som byter namn på eller tar bort en kolumn får de gamla instanser som fortfarande körs att gå sönder, och gör det riskabelt att rulla tillbaka, eftersom den gamla koden räknar med det som togs bort. Dela upp sådana ändringar över flera driftsättningar: lägg först till den nya strukturen, leverera kod som fungerar med båda, och ta bort den gamla strukturen först när ingen körande version behöver den.

Läs mer

- Build, release, run — argumenten i The Twelve-Factor App för tre separata steg och för releaser som aldrig ändras när de väl har skapats.
- Open Container Initiative — projektet inom Linux Foundation bakom specifikationerna för avbildningar, körmiljöer och distribution som de flesta driftsättningar av containrar vilar på.
- OCI Image Format Specification — manifest, lager, konfiguration och de digests av innehållet som identifierar en avbildning.
- Kubernetes Deployments — rullande uppdateringar, strategin recreate, tidsgränsen för framsteg och återgångar.
- Kubernetes images — taggar, digests, policyer för hämtning och varför `:latest` gör det svårare att rulla tillbaka.

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 driftsättning?” 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/deployment/