Engineer

Ansible

et værktøj der sætter en server i en nedskrevet tilstand

Hvad det er

Ansible er open source-software til automatisering, der forbinder til maskiner over SSH og bringer hver af dem i den konfiguration, som et sæt YAML-filer beskriver: hvilke pakker der er installeret, hvilke brugere og filer der findes, hvilke tjenester der kører, og hvordan de er sat op. Maskinen, Ansible køres fra, kaldes kontrolnoden; maskinerne, den konfigurerer, er de styrede noder, og der skal ikke installeres nogen agent på dem, for Ansible logger ind med styresystemets egne legitimationsoplysninger og udfører arbejdet over den forbindelse. Filerne angiver resultatet, og Ansible finder selv ud af på hver vært, om resultatet allerede er opfyldt.

Delene har jævne navne. Et inventory lister de styrede noder og sorterer dem i grupper som `web` eller `db`, med variabler, der gælder for en hel gruppe eller for én vært. En playbook rummer et eller flere plays, og hvert play kobler en gruppe værter til en liste af opgaver. Hver opgave kalder et modul, et lille program, der ved, hvordan én slags ting styres: en pakke, en fil, en tjeneste, en bruger, en firewallregel. Projektets egen dokumentation beskriver playbooks som et gentageligt, genbrugeligt og enkelt system til konfigurationsstyring og udrulning på mange maskiner. En rolle samler opgaver, skabeloner, standardvariabler og handlers under ét navn, så »en PostgreSQL-server« eller »en hærdet SSH-dæmon« skrives én gang og anvendes på alle værter, der har brug for den.

Et kort play viser formen:

```
- name: Web servers serve the site
  hosts: web
  become: true
  tasks:
    - name: nginx is installed
```

```
    ansible.builtin.package:
      name: nginx
      state: present

  - name: nginx runs now and after a reboot
    ansible.builtin.service:
      name: nginx
      state: started
      enabled: true
```

Hver opgave navngiver en tilstand, ikke en handling. De fleste moduler tjekker først, om tilstanden allerede gælder, og afslutter uden at gøre noget, hvis den gør, så anden kørsel af dette play på en konfigureret vært ændrer intet og siger det. Den egenskab, idempotens, er det, der gør det sikkert at køre den samme playbook mod hele flåden hver gang i stedet for at holde regnskab med, hvilken vært der har fået hvilket script.

Den, der overvejer Ansible, sammenligner det som regel med tre naboer. Over for et shellscript er forskellen, at et script lister trin og kører dem igen, uanset om de er nødvendige, mens et modul tjekker, før det handler. Over for Terraform eller Bicep er forskellen laget: de værktøjer opretter cloudressourcer som netværk, databaser og virtuelle maskiner, mens Ansible konfigurerer det, der kører inde i en maskine, når den først findes. Mange teams bruger et af hver. Over for Puppet og Chef er forskellen modellen: de kører som regel en agent på hver maskine, der henter sin konfiguration fra en central server, mens Ansible skubber ud derfra, hvor det startes, og kun kræver SSH i den anden ende.

Brug begrebet i en beslutning

Brug »Hvad er Ansible?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Hele opsætningen skrevet ned som filer, så den kan genskabes præcist og aldrig kun findes i én persons hoved.

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 skrevet 1.000 RBAC-regler til Linux-servere — 1,000 adgangsregler skrevet for at lukke et sikkerhedshul

Har skrevet 1.000 RBAC-regler til Linux-servere Kontakt os →

Hvorfor det betyder noget

En server sat op i hånden er kun så reproducerbar som hukommelsen hos den, der satte den op. Et halvt år senere kan ingen med vished sige, hvorfor en kerneparameter har sin værdi, hvilken pakke der blev låst til en version og

hvorfor, eller hvad der skal ske først, når maskinen genopbygges. Når opsætningen skrives ned som playbooks, bliver den hukommelse til filer, der kan læses, gennemgås i en pull request og køres igen på en ny maskine med samme resultat.

Den skrevne form ændrer også, hvad afvigelse betyder. Når playbooken er beskrivelsen af værten, viser en kørsel i check-tilstand mod en kørende maskine hvert sted, hvor de to er uenige. Med flaget `--diff` viser rapporten de linjer i hver fil, der ville blive ændret. En flåde, der konvergeres jævnligt, holder sig tæt på sin beskrivelse, og en vært, der er rettet i hånden, dukker op som en ændring, næste gang playbooken kører.

Modellen har en pris. Ansible handler kun, når det køres, så afvigelser mellem to kørsler bliver stående til den næste. Opgaver kører i rækkefølge, én ad gangen på tværs af alle matchede værter, og en stor playbook mod mange maskiner er langsommere end et script, der gør én ting. Logik skrevet i YAML og Jinja2-skabeloner bliver svær at følge, når en rolle har fået mange betingelser. Og en kørsel, der stopper undervejs, efterlader sine værter delvist ændret, for Ansible har ingen transaktion at rulle tilbage.

Sådan ser det ud i praksis

Flere projekter på dette site viser, hvordan et nedskrevet miljø ser ud over tid, og hvad der skal til for at holde det ærligt.

Virksomhedens infrastruktur som kode dækker de offentlige tjenester i drift og konfigurationen af andre komponenter, herunder backup- og mailautomatisering, hvis aktivering er et særskilt trin. Den registrerede implementering omfatter 19 playbooks og 34 roller fordelt på 12.065 linjer YAML; tallene beskriver det daværende trin, ikke repositoriets nuværende størrelse. En vært ligger i en tier-gruppe, hvis variabler erklærer, hvilke roller den kører, så provisionering uden argumenter konvergerer hver vært til dens erklærede tilstand. Idempotens behandles som en kontrakt: en konvergeret vært rapporterer nul ændringer, og et play, der ikke kan sige det, er ikke færdigt. Fire kommandonavnerum holder løfterne adskilt, fra en gate, der ikke rører nogen vært, til verifikation, der ændrer en vært med vilje og returnerer en dom. Siden er ærlig om prisen: hver ændring er langsommere at lave end at redigere en fil på serveren, og en konvergering, der halvt gennemføres, er værre end en, der nægter.

Adgangskontrol er et andet sted, hvor en skrevet beskrivelse betaler sig. 1.000 RBAC-regler blev konfigureret med Ansible på tværs af udviklermiljøer, applikationsinstanser, PostgreSQL, MS SQL og Linux-servere, med hver regel knyttet til én rolle og ét system. Det registrerede resultat er et dokumenteret rollebaseret regelsæt og en gentagelig måde at anvende og gennemgå det på. Det er dét, registreringen underbygger; den sætter ikke tal på en reduktion af sikkerhedsrisikoen eller på sparet opsætningstid.

At holde prøvekørslen sandfærdig krævede sit eget arbejde. Check mode blev bragt til at sige sandheden på hele platformen, efter at en prøvekørsel i september 2026 stoppede på en assertion, der tog fejl om værten, og rådede til at sætte et flag, der ville have lempet en sikkerhedssandkasse. En gennemgang fandt seks sonder, der traf beslutning på værdier, værten aldrig havde givet. Sonder, der læser tilstand, som kørslen ikke har rørt, kører nu også under check-tilstand; resten springes over med en besked, der navngiver, hvad der ikke blev verificeret, og reglen kom ind i udviklingsvejledningen.

Hvor det går galt

Det meste besvær med Ansible kommer af at stole mere på en rapport end på mekanismen bag den.

En prøvekørsel melder et rent resultat for kommandoer, den aldrig kørte. Check-tilstand er en simulering. Moduler, der understøtter den, forudsiger deres ændring; dokumentationen siger, at moduler uden understøttelse hverken rapporterer eller gør noget. Modulerne `command` og `shell` understøtter den kun delvist: medmindre `creates` eller `removes` er angivet, springer de kommandoen over og registrerer et resultat med tomt output. Enhver senere `when:`, der læser det resultat, beslutter på en værdi, ingen har målt. En sonde, der kun læser værten, kan få besked på at køre alligevel, og en, der ikke kan køre, bør sige det:

```
- name: Read the firewall state, even in check mode
  ansible.builtin.command: ufw status
  register: fw
  check_mode: false
  changed_when: false

- name: Say what this dry run did not verify
```

```
  ansible.builtin.debug:
    msg: "check mode: the service restart was not exercised"
  when: ansible_check_mode
```

En kørsel stopper på sin niende opgave med de første otte allerede gennemført. Når en opgave fejler på en vært, tager Ansible værten ud af resten af kørslen og lader det, der allerede er ændret, blive stående. Tjek, der kan køre, før noget rører en vært, hører hjemme foran konvergeringen. Det skete efter en ændring af hemmeligheder i virksomhedens eget miljø: et preflight-play validerer nu Vault-indholdet lokalt først og kører de samme opgavefiler som den rigtige konvergering på omkring et sekund, så en hemmelighed, der ikke resolverer, koster en to-sekunders afvisning på en bærbar i stedet for en delvist gennemført ændring på en kørende server. Samme side nævner en fælde: facts sat under ét play overlever det, så et tjek og en konvergering kædet sammen i én kørsel kan dele en værdi, som tjekket burde have læst igen.

En vært, ingen har rørt, melder ændringer ved hver kørsel. Modulerne `command` og `shell` melder en ændring, hver gang de kører, fordi de ikke kan vide, hvad kommandoen gjorde. En playbook fuld af dem konvergerer aldrig til nul, og rigtige ændringer drukner i støjen. Foretræk et modul, der styrer tingen direkte, tilføj `creates` eller `removes`, eller fortæl Ansible med `changed_when`, hvornår output betyder en ændring.

En adgangskode dukker op i en log eller i repositoriet. Variabler er klartekst, medmindre noget krypterer dem. Ansible Vault krypterer variabelfiler eller enkelte værdier, så de kan ligge i repositoriet, og `no_log: true` på en opgave holder dens argumenter og resultater ude af outputtet. Kør playbooken én gang med høj detaljegrad i et testmiljø, og læs, hvad den skriver, før du stoler på nogen af delene.

Læs mere

- Ansible Community Documentation — projektets egen indgang til installation, begreber og alle vejledninger.
- Ansible playbooks — plays, opgaver, moduler, rækkefølgen af udførelse og hvad idempotens betyder i praksis.
- Validating tasks: check mode and diff mode — hvad en prøvekørsel kan og ikke kan se, og kontakterne pr. opgave.
- How to build your inventory — værter, grupper og de variabler, der hører til dem.

- Protecting sensitive data with Ansible Vault — kryptering af filer og enkelte værdier, der ligger ved siden af playbooks.
- ansible.builtin.command — modulets reference, inklusive den delvise understøttelse af check-tilstand.

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 Ansible?« 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/ansible/