Engineer

Podman

en containermotor som ikke trenger noen daemon

Hva det er

Podman er en containermotor for Linux som bygger, henter og kjører de samme OCI-imagene som Docker, men uten en sentral server i midten. Hver `podman` -kommando gjør jobben sin i sin egen prosess og avslutter. En container som er startet, fortsetter å kjøre under en liten tilsynsprosess kalt conmon, og den prosessen er et barn av det som startet den: et skall, et skript eller systemd. Prosjektet beskriver seg selv som daemonløst, med åpen kildekode og bygd for Linux, og sier at de fleste brukere kan lage et alias fra `docker` til `podman` og fortsette som før.

Navnet er en forkortelse for pod manager. En pod, lånt fra Kubernetes, er en gruppe containere som deler et nettverksnavnerom (network namespace) og derfor når hverandre på localhost. Podman kan lage poder for hånd, og `podman kube play` starter containere og poder fra en Kubernetes-fil skrevet i YAML, noe som gjør det til et fornuftig sted å prøve et manifest før det kommer i nærheten av en klynge.

To egenskaper skiller det ut i daglig bruk. Den første er rootless drift, uten root. En vanlig bruker kan kjøre containere, der brukernavnerom (user namespaces) i systemkjernen kobler den brukeren til UID 0 inne i containeren, og brukerens underordnede ID-område (subordinate IDs), fra `/etc/subuid` og `/etc/subgid` , til alle andre UID-er. Nettverkstrafikken for rootless containere går gjennom pasta, en nettverksstakk som kjører i brukerrommet (user space), siden en uprivilegert bruker ikke har rett til å skrive om pakkefilteret på verten.

Den andre er Quadlet. En Quadlet-fil er en kort, deklarativ beskrivelse av en container, et volum, et nettverk, en pod eller et image-bygg, skrevet i den samme ini-syntaksen som en systemd-enhet og lagret under `/etc/containers/systemd/` for hele systemet eller `~/.config/containers/systemd/` for én bruker. Ved oppstart, og

ved hver `systemctl daemon-reload`, gjør en systemd-generator `web.container` om til en vanlig `web.service`. Den eldre kommandoen `podman generate systemd` virker fortsatt, og prosjektet har merket den som foreldet (deprecated) til fordel for Quadlet.

Bruk begrepet i en beslutning

Bruk «Hva er Podman?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Pakket slik at den oppfører seg helt likt på en bærbar PC, på en testserver og i produksjon, og skalerer opp når en travel uke kommer.

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. Driftet 20 Docker-apper på Google Cloud og AWS — 20 containeriserte applikasjoner produksjonssatt og overvåket

Driftet 20 Docker-apper på Google Cloud og AWS Ta kontakt →

Hvorfor det er viktig

Spørsmålet Podman legger på bordet, er hvem som eier prosesstreet. Docker administrerer containere gjennom `dockerd` med egne regler for omstart, egen logging og egne helsesjekker av containerne. På en server som allerede kjører systemd, er det en tjenestestyrer nummer to ved siden av den første, og de to snakker hvert sitt språk.

Med Podman og Quadlet er systemd den eneste tjenestestyreren. En container er en tjeneste som alle andre: `systemctl status` viser den, journald tar vare på utdataene, `After=` og `Requires=` ordner den i forhold til databasen den trenger, og `Restart=` og cgroup-grensene systemd allerede forstår, gjelder for den uten endringer. En driftsperson som kan lese en enhetsfil, kan dermed allerede lese containerlaget.

Det finnes også en sikkerhetsside. Docker-daemonen kjører som root med mindre rootless-modusen er valgt, og Dockers egen sikkerhetsveiledning sier at bare betrodde brukere bør få styre den, fordi en container kan få tilgang til hvilken som helst katalog på verten. Podman holder ingen privilegert daemon i gang mellom kommandoene. Et rootless oppsett snevrer det inn ytterligere: en prosess som bryter ut av containeren sin, havner som en uprivilegert bruker.

Docker mot Podman er derfor mindre en konkurranse om funksjoner enn et spørsmål om hva som passer. Docker er det smidigste valget på en utviklers bærbare maskin med Docker Desktop, for team der arbeidsflyten er bygd rundt Compose, og for verktøy som forventer en Docker-socket, selv om Podman kan

tilby et kompatibelt API når du ber om det. Podman vinner gjerne på servere der systemd allerede er fasiten, og der en daemon som skriver om brannmuren, er en belastning snarere enn en bekvemmelighet.

Slik ser det ut i praksis

Podman bærer containerlaget i to nyere prosjekter, og før det var det feilsøkingsverktøyet for en flåte bygd på Docker.

På én vert kjører containerne som Quadlet-enheter under systemd, med Podman i stedet for Docker. Grunnen er brannmuren: Dockers standardoppførsel publiserer containerporter over reglene herdingsrollen installerer, mens disse containerne bruker vertens nettverk uten noen publiserte nettverksporter i det hele tatt. Enhetene kjører på systemnivå og ikke rootless, og siden fører det opp som en avveining, enkel automatisering mot den strengere isoleringen rootless ville gitt, sammen med et notat om hvilken vei man skal gå hvis utbrudd fra containeren noen gang blir viktigere. Produksjonssetting er skilt fra provisjonering. Root legger stien til rette én gang, og selve utgivelsen er et skript som en uprivilegert bruker får kjøre gjennom en avgrenset regel. Skriptet gjør to ting: bygger imaget og starter tjenesten på nytt.

Det samme oppsettet fungerer på svært lite maskinvare. En vert med én kjerne og 512 MB minne kjører selskapets offentlig tilgjengelige TLS-endepunkter, git-forge og onion-tjeneste, mens automatisering av sikkerhetskopiering er implementert separat. Containerlaget er Quadlet-enheter under den samme tjenestestyreren som alt annet, i stedet for en daemon med sin egen. I løpet av et dokumentert døgn med stikkprøvemålinger falt tilgjengelig minne aldri under omtrent 310 MB av de 464 MB som er brukbare.

Tidligere, med tjue Docker-pakkede applikasjoner spredt over Google Cloud og AWS, gikk feilsøkingen gjennom Podman, mens Kubernetes ble brukt på orkestreringssiden. Imagene var de samme; verktøyet rundt dem var et valg.

Hvor det går galt

Det meste av trøbbelet med Podman kommer av at man venter at det skal oppføre seg som Docker, eller som en vanlig systemd-enhet, når det er litt av begge.

En redigert Quadlet-fil endrer ingenting. Generatoren kjører ved oppstart og ved `systemctl daemon-reload`, så en endring på disken får virkning først når ett av dem skjer. Deretter avviser `systemctl enable` den genererte tjenesten, fordi systemd behandler den som forbigående (transient). Oppstart ved boot hører hjemme i filens egen `[Install]`-seksjon:

```
# /etc/containers/systemd/web.container
[Unit]
Description=Web front end

[Container]
Image=docker.io/library/nginx:1.27
Network=host

[Service]
Restart=always

[Install]
WantedBy=multi-user.target
```

Når filen er lagret, kjører du `systemctl daemon-reload` og `systemctl start web.service`.

En rootless tjeneste stopper når eieren logger ut. Brukerenheter lever inne i brukerens egen systemd-instans, som avsluttes sammen med den siste økten. Kjører du `loginctl enable-linger` for den brukeren, holdes instansen i live fra oppstart, og det trenger enhver rootless tjeneste som skal overleve en omstart.

En rootless webserver ber om nettverksport 80 og blir avvist. Nettverksporter under 1024 krever root eller kapabiliteten `CAP_NET_BIND_SERVICE`, og den regelen gjelder for brukeren Podman kjører som. Publiser en høy nettverksport og sett en proxy foran, eller senk innstillingen `ip_unprivileged_port_start` i systemkjernen med vilje, vel vitende om at den gjelder for alle brukere på verten.

Filene i en bind mount tilhører en fremmed. I rootless modus blir brukerens egen UID til root inne i containeren, og alle andre UID-er i containeren kobles inn i det underordnede ID-området. En prosess som kjører som UID 999 inne i containeren, skriver filer som verten ser som eid av noe i retning av 100998. `podman unshare` åpner et skall inne i den samme koblingen, der `chown` oppfører seg slik containeren forventer, og valget `U` på en montering endrer eierskapet automatisk så det passer containeren.

Les mer

- What is Podman? — prosjektets egen dokumentasjon, med definisjonen av verktøyet som utgangspunkt.
- podman-systemd.unit — oppslagsverket for Quadlet: filtyper, søkestier og hver nøkkel.
- Basic Setup and Use of Podman in a Rootless environment — underordnede ID-er, brukernavnerom og nettverk via pasta.
- podman-unshare — slik går du inn i brukernavnerommet til rootless drift for å undersøke eller rette eierskap.
- podman-generate-systemd — den eldre måten å lage enheter på, og hvorfor den er foreldet til fordel for Quadlet.

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 Podman?» 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/podman/