Engineer

Podman

en containermotor som inte behöver någon daemon

Vad det är

Podman är en containermotor för Linux som bygger, hämtar och kör samma OCI-avbildningar som Docker, men utan någon central bakgrundsprocess i mitten. Varje `podman` -kommando gör sitt arbete i en egen process och avslutas sedan. En startad container fortsätter att köra under en liten vaktprocess som heter conmon, och den processen är ett barn till det som startade den: ett skal, ett skript eller systemd. Projektet beskriver sig som fritt från daemon, byggt på öppen källkod och gjort för Linux, och skriver att de flesta användare kan göra `docker` till ett alias för `podman` och fortsätta som vanligt.

Namnet är en förkortning av pod manager. En pod, lånad från Kubernetes, är en grupp containrar som delar nätverksnamnrymd och därför når varandra på localhost. Podman kan skapa poddar för hand, och `podman kube play` startar containrar och poddar från en YAML-fil för Kubernetes, vilket gör verktyget till ett rimligt ställe att pröva en definition innan den kommer i närheten av ett kluster.

Två egenskaper skiljer det från Docker i det dagliga arbetet. Den första är rootless-drift. En vanlig användare kan köra containrar, där kärnans användarnamnrymder mappar den användaren till UID 0 inuti containern och användarens underordnade ID-intervall, från `/etc/subuid` och `/etc/subgid` , till alla andra UID. Nätverkstrafiken för rootless-containrar går genom pasta, en nätverksstack i användarrymden, eftersom en oprivilegierad användare saknar rätt att skriva om värdens paketfilter.

Den andra är Quadlet. En Quadlet-fil är en kort deklarativ beskrivning av en container, en volym, ett nätverk, en pod eller ett avbildningsbygge, skriven med samma ini-syntax som en systemd-enhet och sparad under `/etc/containers/systemd/` för hela systemet eller `~/.config/containers/systemd/` för

en enskild användare. Vid uppstart, och vid varje `systemctl daemon-reload`, gör en generator i systemd om `web.container` till en vanlig `web.service`. Det äldre kommandot `podman generate systemd` fungerar fortfarande, men projektet har markerat det som föråldrat till förmån för Quadlet.

Använd begreppet i ett beslut

Använd ”Vad är Podman?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Paketerad för att bete sig identiskt på en laptop, på en testserver och i produktion, och för att skala upp när en hektisk vecka kommer.

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. Drev 20 Docker-appar på Google Cloud och AWS — 20 containerbaserade applikationer, driftsatta och övervakade

Drev 20 Docker-appar på Google Cloud och AWS Hör av er →

Varför det spelar roll

Frågan som Podman lägger på bordet är vem som äger processträdet. Docker hanterar containrar genom `dockerd` med egna omstartsregler, egen loggning och egna hälsokontroller av containrarna. På en server som redan kör systemd blir det en andra processhanterare bredvid den första, och de två talar inte samma språk.

Med Podman och Quadlet är systemd den enda processhanteraren. En container är en tjänst som alla andra: `systemctl status` visar den, journald tar emot dess utdata, `After=` och `Requires=` ordnar den i förhållande till databasen den behöver, och `Restart=` och de cgroup-gränser som systemd redan förstår gäller för den utan ändringar. Den som kan läsa en unit-fil kan redan läsa containermiljön.

Det finns också en säkerhetssida. Docker-daemonen körs som root om dess rootless-läge inte har valts, och säkerhetsguiden från Docker själva säger att bara betrodda användare ska få styra den, eftersom en container kan få vilken katalog som helst på värden. Podman håller aldrig en privilegierad daemon igång mellan kommandona. En rootless-uppsättning snävar in det ytterligare: en process som tar sig ut ur sin container hamnar som en oprivilegierad användare.

Docker eller Podman är därför mindre en tävling i funktioner än en fråga om passform. Docker är det smidigare valet på en utvecklares bärbara dator med Docker Desktop, för team vars arbetsflöde är byggt kring Compose och för verktyg som förväntar sig en Docker-socket, även om Podman på begäran kan

erbjuda ett kompatibelt API. Podman brukar vinna på servrar där systemd redan är sanningskällan, och där en daemon som skriver om brandväggen är en belastning snarare än en bekvämlighet.

Så ser det ut i praktiken

Podman bär containermiljön i två nyliga projekt, och innan dess var det felsökningsverktyget för en flotta byggd på Docker.

På en värd körs containrarna som Quadlet-enheter under systemd, med Podman i stället för Docker. Skälet är brandväggen: Docker publicerar som standard containerportar ovanför de regler som härdningsrollen installerar, medan de här containrarna använder värdens nätverk utan några publicerade portar alls. Enheterna körs på systemnivå snarare än rootless, och sidan redovisar det som en avvägning, enklare automatisering mot den striktare isolering som rootless skulle ge, tillsammans med en anteckning om åt vilket håll man ska gå om containerutbrytning någon gång blir viktigare. Driftsättning är skild från provisionering. Root behövs en enda gång, när driftsättningsvägen förbereds. Sedan sköter en oprivilegierad användare varje ny driftsättning med ett fast skript som en avgränsad regel låter den köra: bygg avbildningen, starta om tjänsten, och mer än så tillåter regeln inte.

Samma upplägg fungerar på mycket lite hårdvara. En värddator med en kärna och 512 MB minne driver företagets publika TLS-ändpunkter, git-forge och onion-tjänst, medan automatisering av säkerhetskopiering har implementerats separat. Containerskiktet består av Quadlet-enheter under samma processhanterare som allt annat i stället för en daemon med en egen. Under ett dokumenterat dygns stickprov var det lägsta uppmätta tillgängliga minnet ungefär 310 MB av de 464 MB som går att använda.

Tidigare, med tjugo Docker-paketerade applikationer utspridda över Google Cloud och AWS, gick felsökningen genom Podman, medan Kubernetes användes för orkestreringen. Avbildningarna var desamma, och verktyget runt dem var ett val.

Var det går fel

De flesta problem med Podman kommer av att man väntar sig att det ska bete sig som Docker, eller som en vanlig systemd-enhet, när det är lite av båda.

En ändrad Quadlet-fil ändrar ingenting. Generatorn körs vid uppstart och vid `systemctl daemon-reload`, så en ändring på disk syns först när något av dem har hänt. Sedan vägrar `systemctl enable` att ta den genererade tjänsten, eftersom systemd behandlar den som en transient enhet. Start vid uppstart hör hemma i filens egen sektion `[Install]`:

```
# /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
```

Kör `systemctl daemon-reload` och `systemctl start web.service` när filen är sparad.

En rootless-tjänst stannar när ägaren loggar ut. Användarenheter lever i användarens egen systemd-hanterare, som avslutas med den sista sessionen. Kommandot `loginctl enable-linger` för den användaren håller hanteraren vid liv från uppstart, och det behöver varje rootless-tjänst som ska överleva en omstart.

En rootless-webbserver ber om port 80 och nekas. Portar under 1024 kräver root eller förmågan `CAP_NET_BIND_SERVICE`, och den regeln gäller för den användare som Podman körs som. Publicera en hög port och sätt en proxy framför, eller sänk kärnans inställning `ip_unprivileged_port_start` med avsikt, i vetskap om att den gäller för varje användare på värden.

Filer i en bindmontering tillhör en främling. I rootless-läge blir användarens eget UID root inuti containern, och alla andra UID i containern mappas in i det underordnade intervallet. En process som körs som UID 999 inuti skriver filer som värden ser som ägda av UID 100998 eller däromkring. Kommandot `podman unshare` öppnar ett skal inuti samma mappning, där `chown` beter sig som containern förväntar sig, och alternativet `U` på en montering byter ägare så att den stämmer med containern automatiskt.

Läs mer

- What is Podman? — projektets egen dokumentation, med början i dess definition av verktyget.
- podman-systemd.unit — referensen för Quadlet: filtyper, sökvägar och varje nyckel.
- Basic Setup and Use of Podman in a Rootless environment — underordnade ID, användarnamnrymder och nätverk via pasta.
- podman-unshare — att gå in i användarnamnrymden för rootless-läget för att granska eller rätta ägarskap.
- podman-generate-systemd — det äldre sättet att skapa enheter och beskedet att det är föråldrat till förmån för Quadlet.

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 Podman?” 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/podman/