Engineer

Podman

Container-Engine, die keinen Daemon benötigt

Was es ist

Podman ist eine Container-Engine für Linux, die dieselben OCI-Images wie Docker baut, herunterlädt und ausführt, jedoch ohne einen zentralen Server dazwischen. Jeder `podman` -Befehl erledigt seine Arbeit in einem eigenen Prozess und endet danach. Ein gestarteter Container läuft unter einem kleinen Überwachungsprozess namens conmon weiter. Sein Elternprozess ist das Programm, das ihn gestartet hat: eine Shell, ein Skript oder systemd. Das Projekt bezeichnet sich als daemonlos, quelloffen und für Linux entwickelt. Die meisten Nutzer können laut Projekt `docker` als Alias für `podman` einrichten und weiterarbeiten.

Der Name steht für Pod Manager. Das von Kubernetes übernommene Konzept eines Pods bezeichnet eine Gruppe von Containern, die einen Netzwerk-Namensraum teilen und sich deshalb über localhost erreichen. Podman kann Pods von Hand anlegen, und `podman kube play` startet Container und Pods aus einer Kubernetes-YAML-Datei. Damit eignet es sich dazu, ein Manifest auszuprobieren, bevor es in die Nähe eines Clusters kommt.

Im Alltag zeichnen es zwei Eigenschaften aus. Die erste ist der Rootless-Betrieb. Ein gewöhnlicher Nutzer kann Container ausführen. Dabei bilden die Benutzer-Namensräume des Kernels diesen Nutzer auf UID 0 im Container ab und seinen untergeordneten ID-Bereich aus `/etc/subuid` und `/etc/subgid` auf alle anderen UIDs. Der Netzwerkverkehr von Rootless-Containern läuft über pasta, einen Netzwerkstack im Benutzerraum, weil ein unprivilegierter Nutzer den Paketfilter des Hosts nicht umschreiben darf.

Die zweite ist Quadlet. Eine Quadlet-Datei beschreibt einen Container, ein Volume, ein Netzwerk, einen Pod oder einen Image-Build kurz und deklarativ in derselben ini-Syntax wie eine systemd-Unit. Sie liegt für das System unter

`/etc/containers/systemd/` oder für einen Nutzer unter `~/.config/containers/systemd/` . Beim Systemstart und bei jedem `systemctl daemon-reload` macht ein systemd-Generator aus `web.container` eine gewöhnliche `web.service` . Der ältere Befehl `podman generate systemd` funktioniert weiterhin, wird vom Projekt aber zugunsten von Quadlet als veraltet gekennzeichnet.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Podman?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. So verpackt, dass sie auf dem Laptop, dem Testserver und im Produktivbetrieb gleich funktioniert und bei höherer Auslastung mitwächst.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 20 Docker-Apps auf Google Cloud und AWS betrieben — 20 bereitgestellte und überwachte Containeranwendungen

20 Docker-Apps auf Google Cloud und AWS betrieben Kontakt aufnehmen →

Warum es wichtig ist

Podman wirft die Frage auf, wer für den Prozessbaum verantwortlich ist. Docker verwaltet Container über `dockerd` , mit eigenen Neustartregeln, eigener Protokollierung und eigenen Zustandsprüfungen für Container. Auf einem Server, der bereits systemd verwendet, ist das eine zweite Prozessaufsicht neben der ersten, und beide sprechen nicht dieselbe Sprache.

Unter Podman mit Quadlet ist systemd die einzige Prozessaufsicht. Ein Container ist ein Dienst wie jeder andere: `systemctl status` zeigt ihn an, journald speichert seine Ausgabe, `After=` und `Requires=` ordnen ihn gegenüber der benötigten Datenbank ein, und `Restart=` sowie die systemd bereits bekannten cgroup-Grenzen gelten unverändert. Wer eine Unit-Datei lesen kann, kann bereits die Container-Ebene lesen.

Auch die Sicherheit spielt eine Rolle. Der Docker-Daemon läuft als root, sofern nicht sein Rootless-Modus gewählt wird. Dockers eigener Sicherheitsleitfaden empfiehlt, nur vertrauenswürdigen Nutzern die Steuerung zu erlauben, weil einem Container jedes beliebige Verzeichnis des Hosts übergeben werden kann. Podman hält zwischen Befehlen keinen privilegierten Daemon am Laufen. Eine Rootless-Einrichtung begrenzt die Auswirkungen weiter: Ein Prozess, der aus seinem Container ausbricht, landet als unprivilegierter Nutzer auf dem Host.

Docker oder Podman ist deshalb weniger eine Frage des Funktionsumfangs als der Eignung. Docker passt besser auf einen Entwickler-Laptop mit Docker Desktop, zu Teams mit einem auf Compose aufgebauten Ablauf und zu Werkzeugen, die einen Docker-Socket erwarten. Podman kann allerdings bei Bedarf eine kompatible API bereitstellen. Podman ist häufig die bessere Wahl auf Servern, auf denen systemd bereits die maßgebliche Instanz ist und ein Daemon, der die Firewall umschreibt, eher eine Belastung als eine Erleichterung darstellt.

Wie es in der Praxis aussieht

Podman übernimmt die Container-Ebene in zwei neueren Projekten. Davor diente es als Werkzeug zur Fehlersuche für einen auf Docker aufgebauten Systembestand.

Auf einem Host laufen Container als Quadlet-Units unter systemd, mit Podman anstelle von Docker. Der Grund ist die Firewall: Docker veröffentlicht Container-Ports in seiner Standardkonfiguration vor den Regeln, die die Härtungsrolle installiert. Diese Container verwenden dagegen das Netzwerk des Hosts und veröffentlichen überhaupt keine Ports. Die Units laufen auf Systemebene statt rootless. Die Seite hält das als Abwägung fest: einfachere Automatisierung gegenüber der strengeren Isolation, die Rootless bieten würde. Sie erläutert auch, in welche Richtung es gehen sollte, falls das Risiko eines Container-Ausbruchs einmal schwerer wiegt als die einfachere Automatisierung. Die Bereitstellung ist von der Provisionierung getrennt. Root richtet den Bereitstellungspfad einmalig ein. Jede spätere Veröffentlichung erfolgt über ein festes Skript, das ein unprivilegierter Nutzer aufgrund einer eng begrenzten Regel ausführen darf. Das Skript tut nur zwei Dinge: das Image bauen und den Dienst neu starten.

Auch mit sehr wenig Hardware funktioniert dieselbe Anordnung. Ein Host mit einem Prozessorkern und 512 MB Arbeitsspeicher betreibt die öffentlichen TLS-Endpunkte, die Git-Forge und den Onion-Dienst des Unternehmens. Die Backup-Automatisierung ist separat implementiert. Seine Container-Ebene besteht aus Quadlet-Units unter derselben Prozessaufsicht wie alles andere, anstelle eines Daemons mit einer eigenen. Bei den im Verlauf eines Tages aufgezeichneten Stichproben fiel der verfügbare Arbeitsspeicher nie unter etwa 310 MB der nutzbaren 464 MB.

Zuvor wurden bei zwanzig als Docker-Images verpackten Anwendungen auf Google Cloud und AWS Probleme mit Podman untersucht. Für die Orchestrierung wurde Kubernetes verwendet. Die Images waren dieselben; das Werkzeug für den Umgang mit ihnen ließ sich wählen.

Wo es schiefgeht

Die meisten Probleme mit Podman entstehen, wenn man erwartet, dass es sich wie Docker oder wie eine gewöhnliche systemd-Unit verhält, obwohl es etwas von beidem ist.

Eine bearbeitete Quadlet-Datei ändert nichts. Der Generator läuft beim Systemstart und bei `systemctl daemon-reload`. Eine Änderung auf der Platte bleibt deshalb bis zu einem dieser Ereignisse unsichtbar. Anschließend lehnt `systemctl enable` den erzeugten Dienst ab, weil systemd ihn als transient behandelt. Der Start beim Hochfahren gehört in den eigenen Abschnitt `[Install]` der Datei:

```
# /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
```

Führen Sie nach dem Speichern `systemctl daemon-reload` und `systemctl start web.service` aus.

Ein Rootless-Dienst endet, wenn sich sein Besitzer abmeldet. Benutzer-Units laufen im eigenen systemd-Manager des Nutzers, der mit der letzten Sitzung endet. `loginctl enable-linger` für diesen Nutzer hält den Manager ab dem Systemstart am Leben. Das braucht jeder Rootless-Dienst, der einen Neustart überstehen soll.

Ein Rootless-Webserver verlangt Port 80 und wird abgewiesen. Ports unter 1024 erfordern root oder die Capability `CAP_NET_BIND_SERVICE`. Diese Regel gilt für den Nutzer, unter dem Podman läuft. Veröffentlichen Sie einen hohen Port und schalten Sie einen Proxy davor oder senken Sie bewusst die Kernel-Einstellung `ip_unprivileged_port_start`. Beachten Sie dabei, dass sie für jeden Nutzer auf dem Host gilt.

Dateien in einem Bind-Mount gehören einem Fremden. Im Rootless-Modus wird die eigene UID des Nutzers zu root im Container. Jede andere Container-UID wird auf den untergeordneten Bereich abgebildet. Ein Prozess mit UID 999 im Container schreibt Dateien, die der Host als Eigentum einer UID wie 100998 sieht. `podman unshare` öffnet eine Shell mit derselben Abbildung, in der sich `chown` so verhält, wie der Container es erwartet. Die Mount-Option `U` passt die Eigentümerschaft automatisch an den Container an.

Weiterführende Lektüre

- Was ist Podman? — die eigene Dokumentation des Projekts, beginnend mit seiner Definition des Werkzeugs.
- podman-systemd.unit — die Quadlet-Referenz: Dateitypen, Suchpfade und sämtliche Schlüssel.
- Grundlegende Einrichtung und Nutzung von Podman in einer Rootless-Umgebung — untergeordnete IDs, Benutzer-Namensräume und das Netzwerk mit pasta.
- podman-unshare — den Rootless-Benutzer-Namensraum betreten, um Eigentümerschaft zu prüfen oder zu korrigieren.
- podman-generate-systemd — der ältere Weg zur Erzeugung von Units und seine Kennzeichnung als veraltet zugunsten von Quadlet.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Podman?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/podman/