Rootless Container mit Podman

Kurzvortrag

Aldo Brießmann

02.12.2020

Übersicht

1 Definition Container

2 Container: Pro/Contra

3 Podman vs. Docker

4 Podman verwenden

5 Live-Demo: CodiMD/HedgeDoc

6 Erfahrungsberichte

7 Zukunft

8 Quellen

Disclaimer

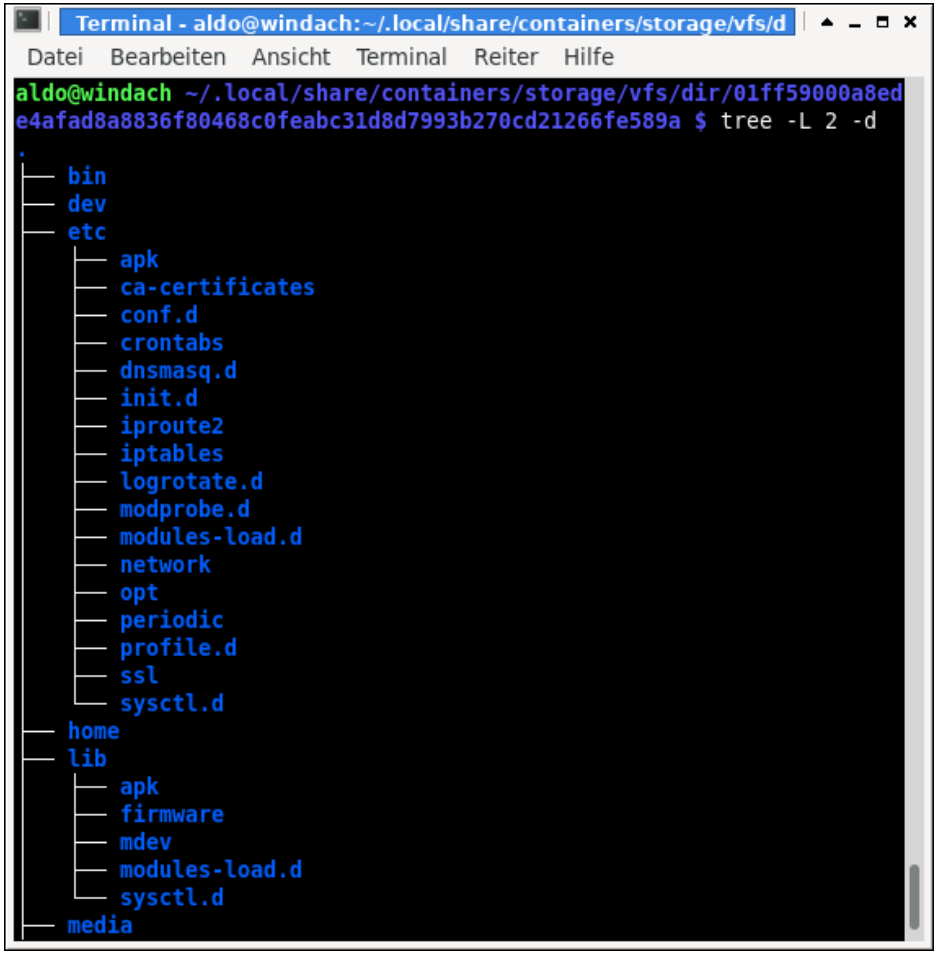
- ich bin nur Privatnutzer
- vor Podman stand ich Containern aufgrund der Sicherheitsbedenken sehr skeptisch gegenüber, jetzt immer noch ein bisschen
- "professioneller"/"industrieller" Containerbetrieb wird mit Kubernetes, OpenShift o.ä. gemacht

was sind Container?

Def. Container(virtualisierung)

Methode, um mehrere Instanzen eines OS isoliert voneinander auf einem Hostsystem zu betreiben

- brauchen weniger Ressourcen als VMs
- aber erfordern Kontrolle über Zugriff auf Hostsystem



Argumente für Container-Einsatz

- weniger Ressourcen-Overhead als bei VMs
- sehr schnell und leicht aufzusetzen
- Neu-Aufsetzen ist automatisch reproduzierbar
- Einsatzgebiete:
  - (Web-)Dienste
  - Build-Systeme (z.B. Paketierung für Linux-Distros, komplexe LaTeX-Dokumente, auch Container-Bau selbst)
- unabhängig vom Host-System
- "kein" Wartungsaufwand klassischer Art (in Configs rumbasteln, ...)
- weit verbreitet, für viele aktuelle Dienste verfügbar (siehe Docker Hub)

Argumente gegen Container-Einsatz

- Abkapselung macht hardwarenahe Aufgaben und grafische Anwendungen schwierig bis unmöglich
- stellenweise komplizierter, schwerer durchschaubar
- ggf. nur mit Zusatzaufwand anzupassen
- weniger Kontrolle bzw. abhängiger von Upstream
- Updates von Dependencies können nicht einfach selbst gemacht werden, ohne Reproduzierbarkeit aufs Spiel zu setzen
  - -> Sicherheitsrisiko[0]
- Sicherheit des Systems und aller Container hängt vom Kernel und von Container-Struktur ab

[0] siehe z.B. `https://vulnerablecontainers.org/`

Wissenswertes zu Docker

- sehr weit verbreitet
- einfach zu benutzen (`docker-swarm`, `docker run`, `docker-compose`, ...)
- viele etablierte Standards, eigenes Ökosystem
- systemweiter Container-Image-Speicher -> keine doppelten Images
- Docker-Daemon (üblicherweise mit Root-Rechten) kümmert sich um Container
- rootless Container in aktuellster Docker-Version verfügbar, aber nur mit Einschränkungen; Docker-Daemon muss von User betrieben werden
- Firma dahinter: Docker Inc.

A brief history of Podman

- um 2015/2016: ein Git-Pull-Request von Redhat-Team an Docker-Projekt wurde abgelehnt
- als Reaktion darauf entstand Skopeo, ein Tool zum Durchsuchen und Verwalten von Container-Images
- ab ca. 2017: Podman als Erweiterung von Skopeo, um Container auch auszuführen
- Mitte 2019: Podman 1.0 veröffentlicht
- Ende 2019: Fedora und OpenSuse wechseln von Docker zu Podman
- Mitte 2020: Podman 2.0 veröffentlicht

Vorteile von Podman

- kein Daemon, stattdessen Prozess-Forks
- Container nicht nur für Root-User bzw. User mit Root-Berechtigungen -> geringeres Sicherheitsrisiko
- gemeinsame Namespaces für mehrere Container in Pods
- Optionale Alternative Docker-Daemon: Podman-Socket
- Systemd für Service-Management in Containern möglich
- separate Kommandozeilen-Tools: `podman-compose` (docker-compose für Podman), `pods-compose` (ähnlich zu docker-compose, aber mit eigener Syntax)
- Container-Image-Speicher per Default pro User (systemweit auch möglich)
- Erstellung von Systemd-Services und K8s-Files für Container
- vereinzelt fehlen noch Docker-Funktionen

Installation

Siehe `https://podman.io/getting-started/installation`

- Debian/Raspbian 10 und andere haben Pakete über OpenSuse Build Service; Ubuntu>=20.10, Suse, Fedora/RHEL, Arch, Gentoo, Void und andere haben eigene Pakete
- für Rootless: SubUIDs und SubGIDs für Benutzer einrichten; braucht einmalig Root-Rechte, leider nur umständlich automatisierbar
  - ENTWEDER: Dateien `/etc/subuid`, `/etc/subuid-`, `/etc/subgid` und `/etc/subgid-` bearbeiten:

```
$ cat /etc/subuid
1000:100000:65536
```

  - ODER: Shell-Befehl usermod nutzen, z.B.

```
$ usermod myuser --add-subuids 100000-165535 \
--add-subgids 100000-165535
```

- regulärer Shell-Betrieb: `podman` als Drop-in-Replacement für `docker`

Live-Demo

Setup von CodiMD/HedgeDoc[1] mittels Podman-Compose
Voraussetzungen:

- Podman Version 2.1.1 (2.x sollte gehen)
- podman-compose Version 0.1.7dev [2]
  Installation:

```
$ pip3 install --user \
https://github.com/containers/podman-compose/archive/devel.tar.gz
```

[1] kollaborativer Markdown-Editor; Umbenennung zu HedgeDoc läuft gerade
[2] offiziell stabil ist gerade noch 0.1.5, die ist aber veraltet

Durchführung

1.
```
$ git clone
https://github.com/hedgedoc/container.git
hedgedoc-container
```
2.
```
$ cd hedgedoc-container
```
3.
```
$ podman-compose up -d
```
(-d für detach)

4. wenn fertig, im Browser `http://localhost:3000/` aufrufen

Auch interessant:

- Systemd-File erstellen: `podman generate systemd --files container`
- Kubernetes-Features: `podman generate kube` und `podman play kube`
- Volumes inspizieren (dort liegen die Daten der Container): `podman volume inspect container_database`

- CodiMD, Greenlight für BigBlueButton, Mautrix-Telegram: jeweils ein Container für Dienst, separate Datenbank, Setup sehr leicht
- Scalelite (BBB Load-Balancer): mehrere Container, komplizierter, aber machbar
- BBB-Node-Exporter (BBB zu Prometheus für Monitoring, nicht vom Internet aus erreichbar): aus Zeitgründen und weil eh Root-Zugriff auf System notwendig war über Docker
- WorkAdventure[3]: momentan noch fehlendes Feature in Podman-Socket-API, bereits in Arbeit
- Discourse: baut finale Dienst-Container in temporärem Container, verwendet dort fest Docker -> mit Podman nicht direkt kompatibel (zuletzt getestet vor mehreren Monaten, noch ohne Podman-Socket)
- Gitlab-Runner mit Podman: bislang Probleme, soll aber demnächst mit Podman 2.2 funktionieren

[3] `https://workadventu.re/`

- 100%-Kompatibilität zu Docker fehlt noch
- einfacher Weg für "Container-Management"
  - starten, stoppen, updaten, separater User für jeden Dienst/Container
  - evtl k3s (Kubernetes in kleiner und einfacher)

- https://de.wikipedia.org/wiki/Containervirtualisierung
- Podman-Dokumentation: https://docs.podman.io/
- Docker-Dokumentation: https://docs.docker.com/
- Gitlab-Runner mit Podman (wohl >=2.2) https://www.redhat.com/sysadmin/history-api
- [5] https://www.netways.de/blog/2019/05/31/podman-ist-dem-docker-sein-tod/
- [5] https://www.heise.de/developer/artikel/Podman-Linux-Container-einfach-gemacht-Teil-1-4329067.html
- [5] https://indico.cern.ch/event/757415/contributions/3421994/attachments/1855302/3047064/Podman_Rootless_Containers.pdf

[5] schon älter, evtl teilweise überholt

Quellen II

- https://balagetech.com/convert-docker-compose-services-to-pods/
- https://rootlesscontaine.rs/
- Container Security 101: https://www.youtube.com/watch?v=QoGNvBHiVcM
- https://vulnerablecontainers.org/
- https://media.ccc.de/v/Camp2019-10178-hacking_containers_and_kubernetes