April 2021

# Rootless Containers With Podman

Or why I have trust issues

Steven Ellis - Red Hat

fossasia
April 2021

## Agenda

What – An overview of the technology

- Containers & Podman

Why rootless

- Should be why wouldn't you run containers rootless

How – Implementing a simple example

- Home Assistant + Mosquitto MQTT

## Container Standards : Runtime interfaces

docker
cri-o

OPEN CONTAINER INITIATIVE

cri-o

## Experience:

- A lightweight, OCI-compliant container runtime designed for Kubernetes
- Runs any OCI compliant, Docker compatible container images
- Improve container security & performance at scale

## Roadmap

- Permanent Kubernetes project
- Continues to track and release with upstream Kubernetes
- On track to become the default container engine for nodes
- Converting node troubleshooting documentation to use crictl for human interface to CRI-O
- Adding user namespace support
- Integrating libpod for better CLI integration with Podman

## Container Standards : Alternative Tooling

docker

podman

skopeo

buildah

OPEN CONTAINER INITIATIVE

# podman

## Experience

- Provides a familiar command line experience compatible with the docker cli
- Great for running, building, and sharing containers outside of OpenShift
- Can be wired into existing infrastructure where the docker daemon/cli are used today
- Simple command line interface, no client-server architecture, so more agile in many use cases

## Roadmap:

- GA in RHEL 7.6 & RHEL 8
    - https://podman.io/getting-started/installation for a wide range of distribution focused guides.
- Run containers as non-root (enhanced user namespaces)
- Docker compatible health checks

buildah

## Experience

- OCI Container images compatible with Docker format
- Multi-stage builds supported with and without dockerfiles
- Customizable image layer caching
- Shares the underlying image and storage components with CRI-O
- Build OCI compatible images as a non-root user

## (don't) get rooted

## Why rootless containers?

We'd mostly solved this on traditional Linux environments

- Apps and services run under "service" userids

Originally all "docker" images had to be run as "root"

```
# docker run -it alpine
```

Rootless containers are containers that can be created, run, and managed by users without admin rights.

Multiple **unprivileged** users can run the same containers on the same machine

## Why Podman?

Fundamentally designed with security in mind

Rootless support built in

Integrates nicely with systemd

Default approach on Fedora and RHEL

## Why Should I Care?

I build my containers from Scratch?

- Really!!.. All of Them?
- Including the Base OS?
- No community containers?
- No 3rd party commercial containers

My container platform is secure

- Really? Good for you!!

We all consume a base OS of some form

- Alpine
- Ubuntu
- UBI8

Growing number of commercial containers

- Microsoft SQL Server has a UBI based container image

## How secure are Docker / k8s

A new security analysis of the 4 million container images hosted on the Docker Hub repository revealed that more than half contained at least one critical vulnerability.

- https://www.csoonline.com/article/3599454/half-of-all-docker-hub-images-have-at-least-one-critical-vulnerability.html
- https://www.securityweek.com/analysis-4-million-docker-images-shows-half-have-critical-vulnerabilities

90% of respondents have experienced a security incident in Kubernetes environments

- https://www.stackrox.com/post/2020/09/top-5-takeaways-from-the-latest-kubernetes-security-report/

Top 5 Kubernetes Vulnerabilities of 2019 – the Year in Review

- https://www.stackrox.com/post/2020/01/top-5-kubernetes-vulnerabilities-of-2019-the-year-in-review/

## Going rootless!

## Be the customer

Validate the technology

- In a way that excites me

Don’t cut corners

- Kinda... Almost

What do **I need** that could/should be in a container?

- Using a 3rd party container.

## re-platform vs net new

Existing Services

- Bunch of websites
- Trac / SVN / Git
- MythTV
- NFS / SMB
- Firewall
- Music Streaming

New and Shiny

- Home Automation
- .....

## Rootless Options

Podman runs as a user "fred"

- Processes inside container run as **root**

```
$ id
uid=1003(fred) gid=1003(fred) groups=1003(fred)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0
:c0.c1023

$ podman pull registry.access.redhat.com/ubi8

$ podman run -it \
registry.access.redhat.com/ubi8 \
/bin/bash

# id
uid=0(root) gid=0(root) groups=0(root)
# whoami
root
```

Podman runs as a user "fred"

- Processes inside run as a **specified user**

```
[fred@pod1 ~]$ podman run -it \
-u nobody \
registry.access.redhat.com/ubi8 \
/bin/bash

bash-4.4$ id
uid=65534(nobody) gid=65534(nobody) groups=65534(nobody)

bash-4.4$ whoami
nobody
```

## Rootless Requirements

Podman 1.6.4 or newer

- Ideally Podman 2.x +

slirp4netns

Increase number of user namespaces

```
# echo "user.max_user_namespaces=28633" > /etc/sysctl.d/userns.conf
# sysctl -p /etc/sysctl.d/userns.conf
```

Additional subordinate SUBIUD/SUBGIUD entries

- Only required if using “system” users
- details provided below in my example

```
cat /etc/subuid /etc/subgid
```

fossasia
April 2021

## HomeAssistant

Many thanks - yet again - to Chris Smart

- https://blog.christophersmart.com/2019/09/20/running-a-non-root-container-on-fedora-with-podman-and-systemd/

Create the user environment

```
useradd -r -m -d /var/lib/hass hass
```

with additional SUBUIDs (if needed)

```
NEW_SUBUID=$(($(tail -1 /etc/subuid \
  |awk -F ":" '{print $2}')+65536))
NEW_SUBGID=$(($(tail -1 /etc/subgid \
  |awk -F ":" '{print $2}')+65536))
sudo usermod \
--add-subuids  ${NEW_SUBUID}-$((${NEW_SUBUID}+65535)) \
--add-subgids  ${NEW_SUBGID}-$((${NEW_SUBGID}+65535)) \
hass
```

Create the config/data directories with the correct SELinux permissions

```
sudo -H -u hass bash -c "mkdir ~/{config,ssl}"
sudo semanage fcontext -a -t user_home_dir_t \
  "/var/lib/hass(/.+)?"
sudo semanage fcontext -a -t svirt_sandbox_file_t \
  "/var/lib/hass/((config)|(ssl))(/.+)?"
sudo restorecon -Frv /var/lib/hass
```

Expose the service

```
firewall-cmd  --add-port=8123/tcp --permanent
```

# Hass container

## Initial testing

```
podman run -dt \
--name=hass \
-v /var/lib/hass/config:/config \
-v /var/lib/hass/ssl:/ssl \
-v /etc/localtime:/etc/localtime:ro \
--net=host \
docker.io/homeassistant/home-assistant:latest

podman ps -a
```

## Check the service is running

```
podman logs hass
```

## Enable as systemd service

```
cat << EOF | sudo tee /etc/systemd/system/hass.service
[Unit]
Description=Home Assistant in Container
After=network.target

[Service]
User=hass
Group=hass
Type=simple
TimeoutStartSec=5m
ExecStartPre=-/usr/bin/podman rm -f "hass"
ExecStart=podman run --name=hass -v
/var/lib/hass/ssl:/ssl:ro -v /var/lib/hass/config:/config
-v /etc/localtime:/etc/localtime:ro --net=host
docker.io/homeassistant/home-assistant:latest
ExecReload=-/usr/bin/podman stop "hass"
ExecReload=-/usr/bin/podman rm "hass"
ExecStop=-/usr/bin/podman stop "hass"
Restart=always
RestartSec=30

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

## MQTT

Need a mqtt broker to handle some of my devices

- mosquitto mqtt is a perfect fit

Test run as hass user

```
podman run --name mosquitto \
  --rm -p "9001:9001" -p "1883:1883" \
        eclipse-mosquitto:latest
```

Enable as systemd service

```
cat << EOF | sudo tee /etc/systemd/system/mosquitto.service
[Unit]
Description=Home Assistant in Container
After=network.target

[Service]
User=hass
Group=hass
Type=simple
TimeoutStartSec=5m
ExecStartPre=-/usr/bin/podman rm -f "mosquitto"
ExecStart=podman run --name mosquitto \
  --rm -p "9001:9001" -p "1883:1883" \
  eclipse-mosquitto:latest
ExecReload=-/usr/bin/podman stop "mosquitto"
ExecReload=-/usr/bin/podman rm "mosquitto"
ExecStop=-/usr/bin/podman stop "mosquitto"
Restart=always
RestartSec=30

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

## Good/Bad/Frustrating

Frustrating

- Initial rootless support in RHEL8.1 podman wasn't fully functional
    - Weird memory errors running hass
    - Tested an early engineering build of podman to validate and resolve
    - No issues as of GA RHEL 8.2
- Would have been painless on Fedora

Bad

- Not all containers are ready to be rootless
    - It isn't easy to identify
    - Your mileage may vary
    - Many need to run as root inside the container
- Crash consistency issues
    - Appears to be a lot better with more recent podman builds
    - Previously had to manually clean up dead pods.

Good

- Very easy to update the service
- Configuration and Data are very easy to back/migrate
- I "feel" safer.

## Troubleshooting

Very similar to docker troubleshooting

Check for old/dead images

```
podman ps -a

podman logs <image_name>
```

Stop and cleanup old/dead images

```
podman stop <image_name>

podman rmi <image_name>
```

If you're using systemd - avoid starting images manually if possible

```
systemctl stop hass
systemctl stop mosquitto

systemctl start hass
systemctl start mosquitto
```

## Upgrading

Pull the new image in advance as the required user

```
# su - hass
$ podman pull   eclipse-mosquitto:latest
```

Restart the service using systemd

```
# systemctl stop mosquitto

# systemctl start mosquitto
```

## References

12 Podman guides to get started with containers

Rootless containers with Podman: The basics

What happens behind the scenes of a rootless Podman container?

Rootless containers using Podman – Video Series

Experimenting with Podman

## Questions?

sellis@redhat.com
http://people.redhat.com/sellis