THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
September 2021

Rootless Containers With Podman

Or why I have trust issues

Steven Ellis - Red Hat

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT

Red Hat

Agenda

What - An overview of the technology

- Containers & Podman

Why rootless

- And why you should care

How - Implementing a simple example

- Home Assistant + Mosquitto MQTT
- Minidlna + NFS

Container Standards : Runtime interfaces

docker → cri-o

OPEN CONTAINER INITIATIVE

THE LINUX FOUNDATION OPEN SOURCE SUMMIT September 2021

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

Red Hat

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
- As of podman 2.0, API server compatible with docker API

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, leveraging SELinux

Smaller attack surface - just a runtime engine

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
- RHEL ubi8

Growing number of commercial containers

- Microsoft SQL Server has a UBI based container image

How secure are Docker / k8s

A recent 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

94% of respondents have experienced a security incident in Kubernetes environments

- https://www.redhat.com/en/resources/state-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
- minidlna

New and Shiny

- Home Automation
- .....

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
```

Podman Details

Confirm version of podman available

```
# podman version
Version:       3.2.3
API Version:   3.2.3
Go Version:    go1.15.7
Built:         Tue Jul 27 19:29:39 2021
OS/Arch:       linux/amd64
```

Latest Podman running on RHEL 8.4

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/ubi-micro

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

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

bash-4.4$ whoami
nobody
```

Podman basics

Podman is interchangeable with Docker as an container engine

- runC is the default OCI compatible container runtime

```
$ podman images
REPOSITORY                                  TAG         IMAGE ID      CREATED       SIZE
registry.access.redhat.com/ubi8/ubi-micro   latest      c5ba898d3645  3 weeks ago   38.9 MB

$ podman ps


$ podman ps -a
CONTAINER ID  IMAGE                                             COMMAND    CREATED             STATUS
PORTS       NAMES
afd03a05d62d  registry.access.redhat.com/ubi8/ubi-micro:latest  /bin/bash  About a minute ago  Exited (0) About a
minute ago            gifted_blackburn
0930c4f9d9fb  registry.access.redhat.com/ubi8/ubi-micro:latest  /bin/bash  About a minute ago  Exited (0) About a
minute ago            brave_haibt
```

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
firewall-cmd --reload
```

Hass container

Initial testing

```
# su - hass
$ 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
```

Check the service is running

```
$ podman logs hass
```

Cleanup

```
$ podman stop hass; podman rm 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 running Tasmota.

TASMOTA

Mosquitto mqtt is a perfect fit and has an "off the shelf" container image

Test run as hass user

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

Updated MQTT

New mosquitto builds now **requires** a config file to match my environment

```
$ mkdir mosquitto
$ cat << EOF | tee mosquitto/mosquitto.conf
listener 1883
allow_anonymous true
EOF
```

Re-test

```
podman run --name mosquitto \
  --rm -p "9001:9001" -p "1883:1883" \
 -v
"/var/lib/hass/mosquitto/mosquitto.conf:/mos
quitto/config/mosquitto.conf:Z" \
        eclipse-mosquitto:latest
```

Include config file in the 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" \
  -v
"/var/lib/hass/mosquitto/mosquitto.conf:/mosquitto/config/mos
quitto.conf:Z" eclipse-mosquitto:latest \
  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
```

What Next?

Minidlna in a container has non-root issues with NFS bases volumes

- Audio / Video storage is on NFS

```
$ podman volume create --opt type=nfs --opt o=ro --opt device=svr:/opt/VideoVol VideoVol
$ podman volume create --opt type=nfs --opt o=ro --opt device=svr:/opt/Audio Audio
$ podman volume ls
DRIVER      VOLUME NAME
local       Audio
local       VideoVol
```

- Testing via the UBI8 image fails

```
$ podman run -v VideoVol:/mnt/VideoVol -it registry.access.redhat.com/ubi8-micro /bin/bash
Error: error mounting volume VideoVol for container
5c56d9e29821cc494a6f5621513222a8f7ee98a07967ad5665de5daa6c5b9f54: cannot mount volumes
without root privileges: operation requires root privileges
```

Good/Bad/Frustrating

Frustrating

- Initial rootless podman support in RHEL8.1 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
- Podman issues managing NFS volumes as a non-root user

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 backup/migrate
- I "feel" safer.
- A lot more lightweight than multiple VMs.

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 rm <old_instance>

podman rmi <image_name>

podman system prune
```

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

```
systemctl stop hass
systemctl stop mosquitto

systemctl start hass
systemctl start mosquitto
```

Upgrading Workloads

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
```

Upgrading Podman

Podman's system command has various maintenance options

```
$ podman system --help
Manage podman

Description:
  Manage podman

Usage:
  podman system [command]

Available Commands:
  connection  Manage remote ssh destinations
  df          Show podman disk usage
  info        Display podman system information
  migrate     Migrate containers
  prune       Remove unused data
  renumber    Migrate lock numbers
  reset       Reset podman storage
  service     Run API service
```

If you've performed a major Podman upgrade run the following

```
$ podman system migrate
```

And if you're still experiencing issues try

```
$ podman system reset
```

Podman Maintenance

Podman's system command can also clean up your environment.

```
$ podman system prune
WARNING! This will remove:
        - all stopped containers
        - all networks not used by at least one container
        - all dangling images
        - all dangling build cache

Are you sure you want to continue? [y/N] y
podman Deleted Containers
0930c4f9d9fb137eee7691097ba22e798197e624d4e65a6699b914ad1f6d7791
afd03a05d62de2a98e1f98529fd02f8b1ebb9a53e437e0a328ea2a2e833e2489
Deleted Images
Total reclaimed space: 27B
$ podman ps -a
CONTAINER ID  IMAGE       COMMAND     CREATED     STATUS      PORTS       NAMES
$ podman images
REPOSITORY                                TAG         IMAGE ID      CREATED      SIZE
registry.access.redhat.com/ubi8/ubi-micro  latest      c5ba898d3645  3 weeks ago  38.9 MB
```

References

Getting Started with Podman

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

Podman Katacoda Tutorial

THE LINUX FOUNDATION OPEN SOURCE SUMMIT

Red Hat

Questions?

THE LINUX FOUNDATION

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

Questions?

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

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT

Red Hat

Click to Edit Title

Name / Subtitle Here

#ossummit @twitterhandle

THE LINUX FOUNDATION

Click to edit title

- Click to edit text
  - Second level
    - Third level
      - Fourth level
        - » Fifth level

Click to place text here