Red Hat
Training and
Certification

Managing and Building Container Images and Containers

Travis Michette

Version 1.0

Table of Contents

1. Managing Containers with the New Runtime

1.1. Deploying Containers with the New Container Runtime

1.1.1. The Podman Container Engine

RHEL8 includes he **container-tools** package module. New engine is **podman** replaces **docker** and **moby**. It also contains new tools **buildah** to build container images and **skopeo** to manage images on registries like **runc**. The new toolset allows building/running containers without daemons.

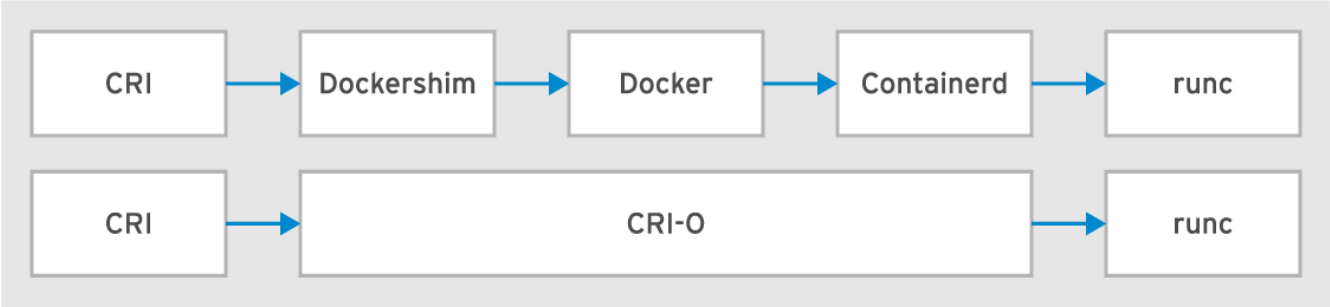

Figure 1. Docker to RHEL8 Container Runtime

Container Runtime Toolset

- Docker replaced with new container runtime
- New toolset supports OCI and reuse of third-party images
- Integrates with **audit** of Docker client-server model
- **container-tools** module provides new container runtime tools and engine.

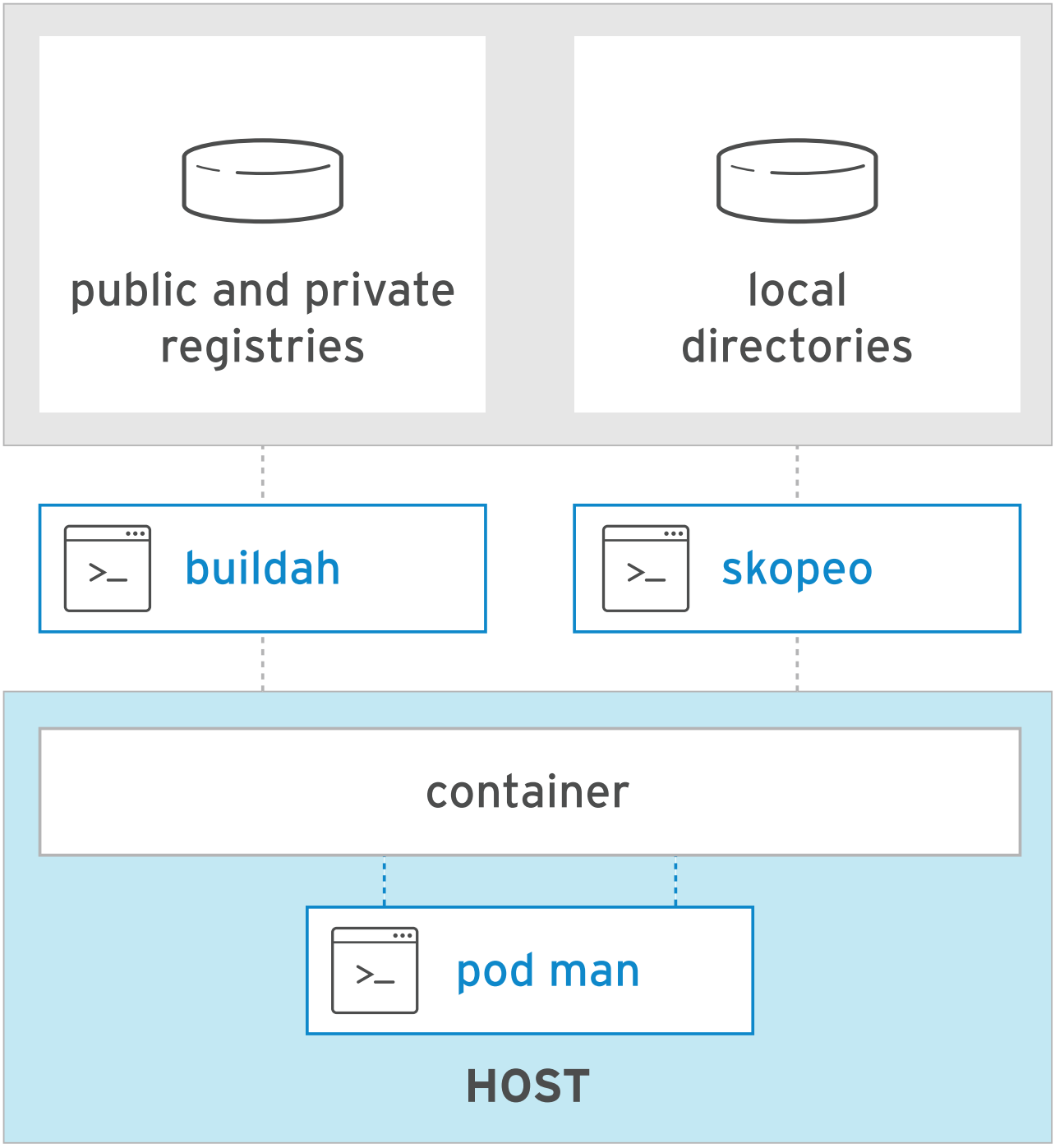

Figure 2. New Container Runtime

Describing new Container Runtime Tool

- The **podman** engine is daemonless and supporting container execution.
- **podman** syntax is similar to the docker command, supporting **Dockerfile** use
- **Buildah** builds container images, from scratch or a Dockerfile.
- Copy and inspect container images in registries with **Skopeo**
- **Skopeo** supports Docker and private registries, the Atomic registry, and local directories, including those which use OCI

RHEL8 includes **Pacemaker** containers with **podman** as a tech preview. Pacemaker supports execution of the container across multiple hosts.

Example 1. Using Container Tools

Listing 1. Installation of Container Tools

```
[student@workstation ~]$ sudo yum module install container-tools
```

Listing 2. Creating a Custom Container

```
[student@workstation ~]$ buildah from scratch
working-container
```

Listing 3. Naming and Inspecting a Custom Container

```
[student@workstation ~]$ buildah config --label name=My-Container working-container
[student@workstation ~]$ buildah inspect working-container
```

Listing 4. Installing Packages on Working Container

```
[student@workstation ~]$  buildah mount working-container ①


[student@workstation ~]$ yumdownloader --destdir=/tmp redhat-release-server ②

[student@workstation ~]$ rpm -ivh --root
/var/lib/containers/storage/overlay/a6a136063f0ada2b1ed4b01eff9a04b4d6419ae828bc4b49e742bca594e08560/merged /tmp/redhat-release-8.0-
0.39.el8.x86_64.rpm ③

[student@workstation ~]$ cp  /etc/yum.repos.d/rhel_dvd.repo
/var/lib/containers/storage/overlay/a6a136063f0ada2b1ed4b01eff9a04b4d6419ae828bc4b49e742bca594e08560/merged/etc/yum.repos.d/ ④

[student@workstation ~]$ yum install --installroot
/var/lib/containers/storage/overlay/a6a136063f0ada2b1ed4b01eff9a04b4d6419ae828bc4b49e742bca594e08560/merged httpd ⑤


[student@workstation ~]$ echo "This is a custom webserver container for me" >>
/var/lib/containers/storage/overlay/a6a136063f0ada2b1ed4b01eff9a04b4d6419ae828bc4b49e742bca594e08560/merged/var/www/html/index.html ⑥

[student@workstation ~]$  yum install --installroot
/var/lib/containers/storage/overlay/a6a136063f0ada2b1ed4b01eff9a04b4d6419ae828bc4b49e742bca594e08560/merged httpd-manual ⑦

[student@workstation ~]$ buildah config --cmd "/usr/sbin/httpd -DFOREGROUND" working-container ⑧

[student@workstation ~]$ buildah config --port 80/tcp working-container ⑨

[student@workstation ~]$ yum clean all --installroot
/var/lib/containers/storage/overlay/a6a136063f0ada2b1ed4b01eff9a04b4d6419ae828bc4b49e742bca594e08560/merged ⑩

[student@workstation ~]$  buildah unmount working-container ⑪

[student@workstation ~]$ buildah commit working-container my-container-image ⑫

[student@workstation ~]$ buildah images ⑬
```

① Mount container image filesystem for modification

② Download Red Hat Release RPM for installation

③ Install Red Hat Release RPM

④ Create repository for container image so files can be installed

⑤ Install the HTTP package for a webserver

⑥ Create an **index.html** file for the webserver

⑦ Install the Apache manual for reference documentation

⑧ Configure webserver to run

⑨ Configure and open port **80** for the **TCP** protocol for the container

⑩ Clean up yum data to minimize required disk space

⑪ Unmount the container image filesystem

⑫ Commit the container image

⑬ List container images

Listing 5. Testing the Container Image

```
[student@workstation ~]$ podman run -d -p 8080:80 localhost/my-container-image

[student@workstation ~]$ curl localhost:8080
This is a custom webserver container for me

[student@workstation ~]$ curl http://localhost:8080/manual/
```

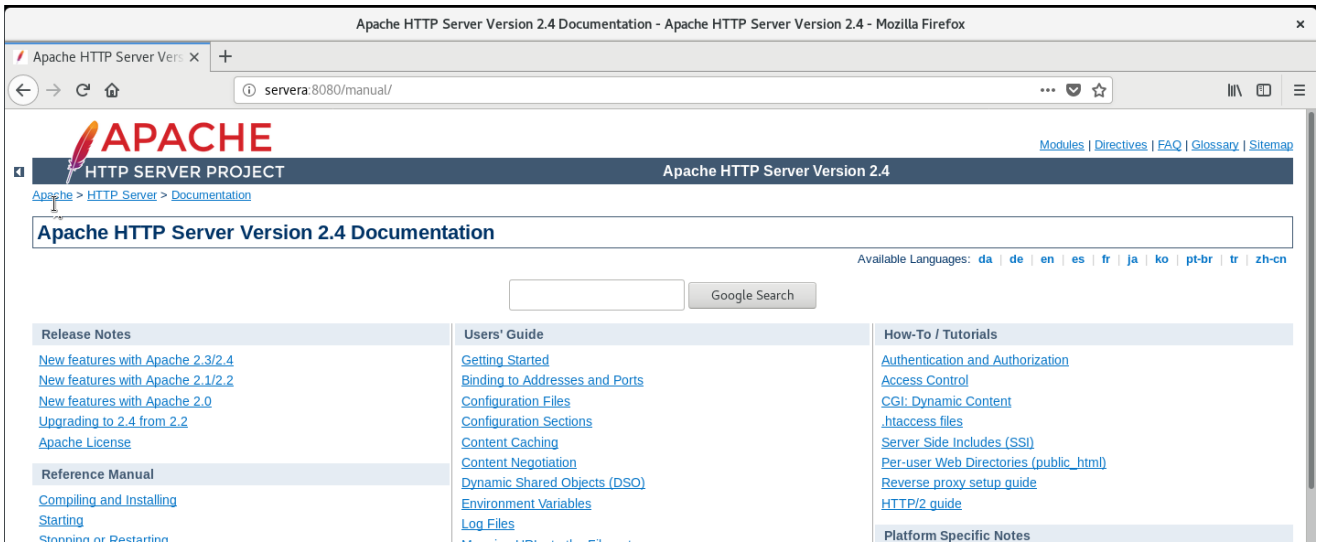

Figure 3. Testing Container

Listing 6. Stopping and Cleanup of Image

```
[student@workstation ~]$ podman list ①
CONTAINER ID  IMAGE                                COMMAND               CREATED        STATUS            PORTS                 NAMES
9bd572633953  localhost/my-container-image:latest  /usr/sbin/httpd -...  2 seconds ago  Up 1 second ago  0.0.0.0:8080->80/tcp
cranky_stonebraker

[student@workstation ~]$ podman stop 9bd572633953 ②
9bd572633953276ac75417db3ac8e70875a0f2713e8cdfd32253fe343d06153d

[student@workstation ~]$ podman stop -a ③

[student@workstation ~]$ podman rm cranky_stonebraker ④

[student@workstation ~]$ podman rm 9bd572633953276ac75417db3ac8e70875a0f2713e8cdfd32253fe343d06153d ⑤

[student@workstation ~]$ podman rmi localhost/my-container-image ⑥

[student@workstation ~]$ buildah delete working-container ⑦
```

① Listing Running Containers

② Stopping Single Container by ID

③ Stopping All Running Containers

④ Remove Container by Name

⑤ Remove Container by ID

⑥ Removing Container Image from Registry

⑦ Delete Working Container from System