

Let's meet Podman

Features and benefits of Podman and its companion tools

Alessandro Arrichiello
Senior Solution Architect

Gianni Salinetti
Senior Solution Architect

DEVELOPERS

Join the Red Hat developers community

Built for developers, by developers—Red Hat® Developers provides access to software, tools, guides, and resources to make your job and daily work easier. We cover topics that are important to you, like microservices, Linux™ containers, cloud-native apps, Kubernetes, Istio, service mesh, reactive programming, and a lot more.

Join now for free

<https://developers.redhat.com>

Products

- RHEL & RHEL for SAP
- Red Hat OpenShift Container Platform
- Red Hat OpenShift Cloud Functions
- Red Hat build of OpenJDK for RHEL and Windows
- Red Hat build of Quarkus
- RHAMP
- Red Hat Decision Manager
- Red Hat AMQ
- Red Hat Data Virtualization
- Red Hat Fuse
- Red Hat EAP
- Red Hat Mobile Application Platform
- Red Hat Process Automation Manager
- Red Hat Runtimes
- Red Hat SSO
- Red Hat Ansible Automation

Developer Tools

- CDK/CodeReady Containers (OpenShift for your laptop)
- CodeReady Studio
- CodeReady Workspaces


Information & Education

- Red Hat Customer Portal access
- Exclusive Summit & AnsibleFest content for developers
- RHD Reference Materials
 - Books, cheat sheets, webinars, etc.

What's not included currently:

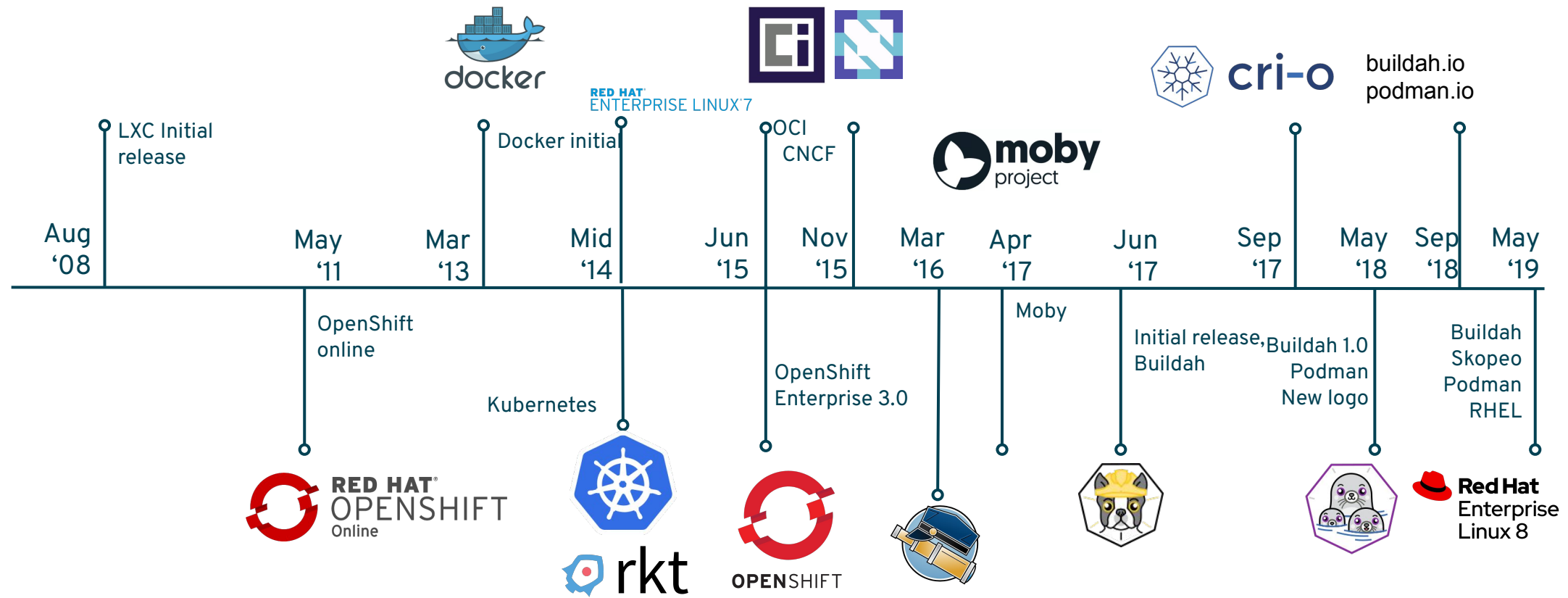
- Support for any tools or products

Agenda

- Containers History and Runtimes
- Podman architectural deep dive
- Running rootless containers with Podman
- Building images with Buildah
- Manipulating images with Skopeo
- Integration with systemd, docker-compose and Kubernetes
- Interactive Labs
- Further Readings
- Quiz and Awards 

Containers Evolution: From Docker to Podman

Container innovation continues



Introducing Podman and its companion tools

Podman, Buildah and Skopeo



Podman

- the POD MANager - is a tool for managing containers and images, volumes mounted into those containers, and pods made from groups of containers.



Buildah

is a tool that facilitates building Open Container Initiative (OCI) container images without a full container runtime or daemon installed

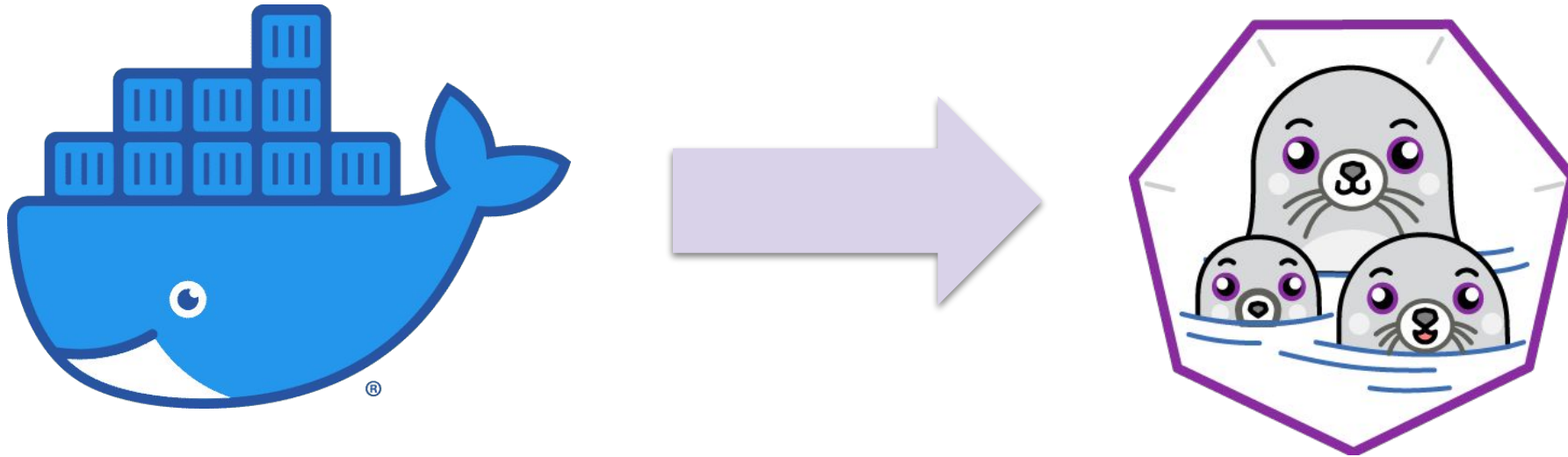


Skopeo

works with remote images registries - retrieving information, images, signing content

From Docker to Podman

Helping the transition between technologies



Podman is designed to provide a smooth transition experience to Docker users thanks to CLI compatibility and OCI compliance.

However, knowing the technology foundations and main differences between the two engines is useful to provide a better migration experience.

Main differences between Docker and Podman

Overview of the main technology pillars

Docker is a daemon-based container engine that consists of three fundamental pillars:

- ▶ **Docker daemon**, a service running in background that supervises the containers orchestration with a client/Server model
- ▶ **Docker REST API**, an interface to interact with the daemon
- ▶ **Docker CLI**, a CLI interface to manage containers and related resources

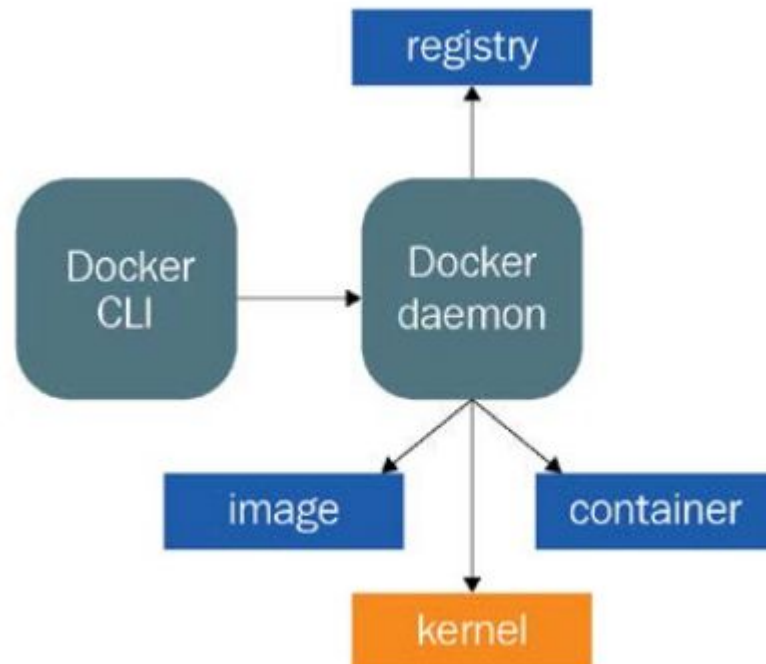
Podman is a container engine that enables users to manage containers, images, as well as storage and network resources:

- ▶ **Daemonless** architecture with fork/exec model
- ▶ Standard **libpod** library for container lifecycle
- ▶ Native support for **rootless** containers and Kubernetes **pods**
- ▶ Full support for OCI runtime/images/storage specs

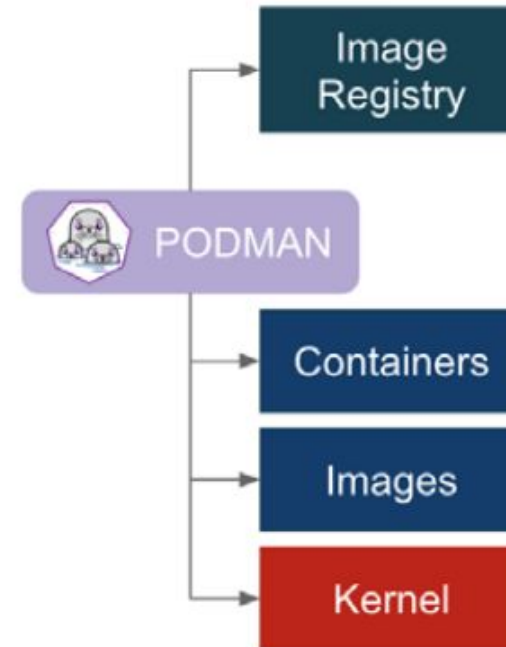
A 10000 ft view

An high level overview of the two different architectures

Docker



Podman



sed 's/docker/podman/g'

Different engines, same driving experience

Podman provides a compatible command line interface for most of Docker standardized commands.

Simply swap commands or use podman-docker default aliasing.

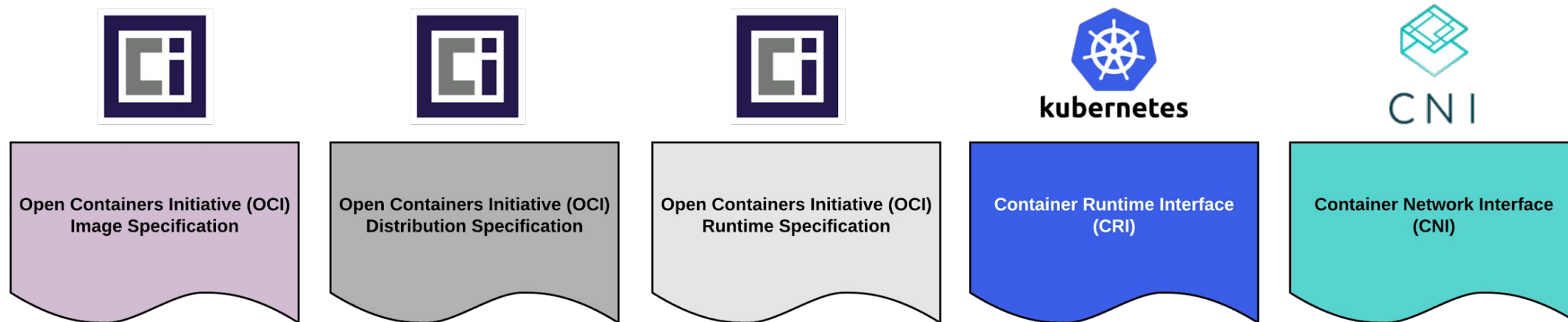
```
# Running an Nginx container with Docker  
$ docker run -d --rm docker.io/library/nginx
```

```
# Running an Nginx container with Podman  
$ podman run -d --rm docker.io/library/nginx
```

Podman architectural deep dive

Cloud-Native standards compliance

Podman and its related components implement multiple standards of the cloud native ecosystem.

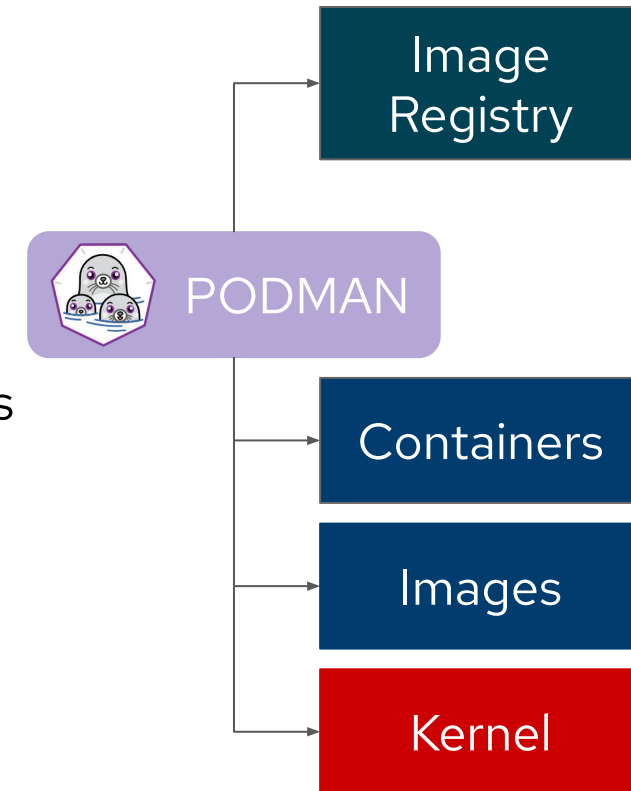


Podman Features

The new container CLI



- A daemon-less CLI/API for running, managing, and debugging OCI containers and pods
- Fast and lightweight
- Leverages runC and crun as container runtimes
- Provides a “Docker-like” syntax for working with containers
- Choose between standard CNI networking or Netavark (Podman 4.x)
- Provides systemd integration, advanced namespace isolation and support for CGroups v2
- Shares state with CRI-O and with Buildah!
- Remote management Varlink / Restful API



SECURITY FEATURES

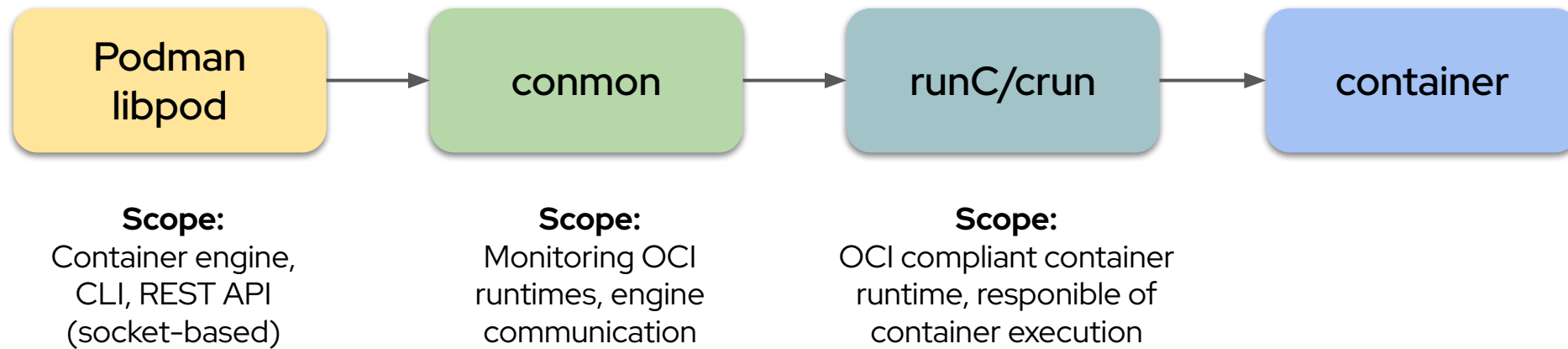
Run and develop securely
No daemon
Run without root
Isolate with user namespaces
Audit who runs what



Container execution workflow

A graphical representation

The following diagram show the logical workflow, from Podman execution to the running container. Podman uses traditional **fork/exec** model for container execution which allows better security through audit logging.



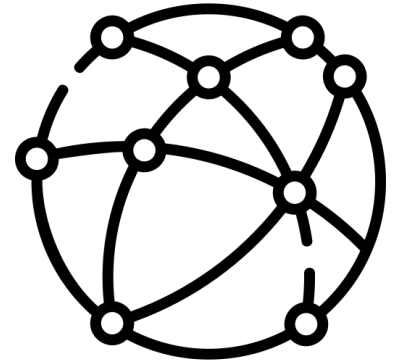
Revamped Network Stack

Introducing Netavark

Podman 4.0 introduced a new default network stack: **Netavark**

Netavark is a **Rust** based network stack for containers. It is being designed to work with Podman but is also applicable for other OCI container management applications.

Users can still choose to operate with the legacy CNI plugin.



Running containers and pods

Pods as minimal execution units

Podman supports the execution of containers and **pods**. A pod is a single execution unit where multiple containers can share Linux namespaces.

```
# Initializing an empty pod
$ podman pod create --name wp-pod

# Creating containers inside the pod
$ podman create --pod wp-pod --name db -d docker.io/library/mysql
$ podman create --pod wp-pod --name wp -d docker.io/library/wordpress

# Running the pod
$ podman pod start wp-pod
```

Running rootless containers with Podman

Rootless containers with Podman

Advantages



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

Rootless containers have several advantages:

- ▶ Additional security layer: even if the container engine, runtime, or orchestrator is compromised, the attacker won't gain root privileges on the host.
- ▶ Multiple unprivileged users to run containers on the same machine
- ▶ Better isolation inside of nested containers (useful for containerized builds)

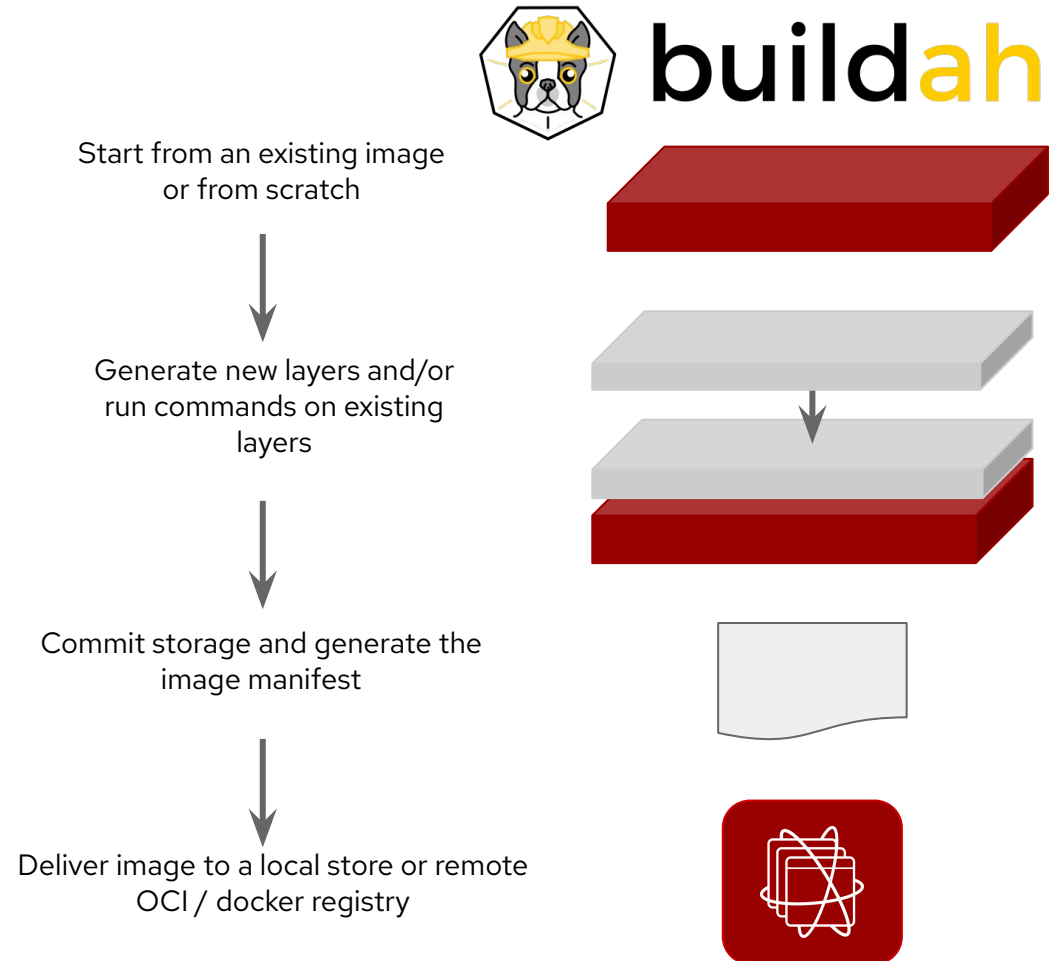
Building images with Buildah

Introducing Buildah

- Now buildah.io
- Builds OCI compliant images
- No daemon - no "docker socket"
- Does not require a running container
- Can use the host's user's secrets.
- Single layer, from scratch images are made easy and it ensures limited manifest.
- If needed you can still maintain Dockerfile based workflow

SECURITY FEATURES

Build securely
No daemon
Shrink the attack surface
Fine-grained control of the layers
Run builds isolated
Better secret management





Buildah native build approach

Using Buildah's command line for builds

Buildah implements Dockerfile instructions as CLI commands allowing more control on builds using scripts, automations and pipelines.

```
$ container=$(buildah from fedora)
$ buildah run $container -- dnf install -y httpd; dnf clean all
$ buildah config --cmd "httpd -DFOREGROUND" $container
$ buildah config --port 80 $container
$ buildah commit $container myhttpd
$ buildah tag myhttpd registry.example.com/myhttpd:v0.0.1
$ buildah push registry.example.com/myhttpd:v0.0.1
```



Building from scratch

Easily create custom distroless images

Buildah supports builds from scratch, (ie from an empty directory structure), even in rootless mode. This feature allows users to create their own custom minimal or distroless images.

```
$ buildah from scratch
$ buildah unshare
# scratchmount=$(buildah mount working-container)
# dnf install --installroot $scratchmount --releasever 34 \
  bash coreutils --setopt install_weak_deps=false -y
$ buildah unmount working-container
$ buildah commit working-container custom-fedora
$ buildah rm working-container
```

Manipulating images with Skopeo

Introducing Skopeo

features and requirements



Skopeo is a command line utility that performs various operations on container images and image repositories:

- ▶ Copying an image from and to various storage mechanisms
- ▶ Inspecting a remote image showing its properties including its layers
- ▶ Deleting an image from an image repository

It works with API V2 container image registries such as *docker.io* and *quay.io* registries, private registries, local directories and local OCI-layout directories. Here some requirements:

- ▶ does not require the user to be running as root to do most of its operations
- ▶ does not require a daemon
- ▶ `sudo dnf -y install skopeo`

Inspecting an image

`skopeo inspect`



Skopeo is able to inspect a repository on a container registry and fetch images layers. The **inspect** command fetches the repository's manifest and it is able to show you a docker inspect-like json output about a whole repository or a tag.

```
$ skopeo inspect docker://registry.access.redhat.com/ubi8/ubi:8.5-236.1648460182
{
  "Name": "registry.access.redhat.com/ubi8/ubi",
  "Digest": "sha256:f20fb774c96377b793475021aca89909da74e9da05136eb6e824aa16f85f22db",
  "RepoTags": [
    "8.4-213",
    ...
  ],
  "Labels": {
    "architecture": "x86_64",
    "build-date": "2022-03-28T10:36:18.413762",
    ...
  }
}
```

Copying an image

skopeco copy



Skopeo can copy container images between various storage mechanisms

```
$ skopeo copy docker://registry.access.redhat.com/ubi8/ubi:8.5-236.1648460182 dir:/tmp/ubi
Getting image source signatures
Checking if image destination supports signatures
Copying blob 4eef1fa1f1c1 done
Copying blob eb24191cef20 done
Copying config c54243b588 done
Writing manifest to image destination
Storing signatures

$ ls /tmp/ubi
4eef1fa1f1c17f9ad6a8187dd5a483e11bd340fc116057bd4cece92305151072  manifest.json  signature-3
signature-6
c54243b58814cd424740dfebb046f356ba3acc23f04e04ffba60004eb1e8b0ea  signature-1    signature-4
version
eb24191cef200934f4fe601f3dc7e7847ea86ce5c52cb9859361bf8e00fe95e7  signature-2    signature-5
```

Integration with systemd, docker-compose and Kubernetes

Podman and systemd

Podman generate systemd

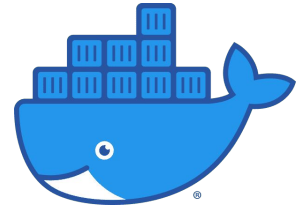


Podman provides a native integration with **Systemd** to manage automatic startup of containers as systemd services, control start-up order, dependency checking, failed service recovery and images auto updates.

```
# Creating a new container
$ podman run -d --name nginx-svc nginx:latest

# Generating a systemd unit
$ podman generate systemd nginx-svc | sudo tee /etc/systemd/system/nginx-svc.service

# Reloading systemd units
$ systemctl daemon-reload
```



Podman and Docker Compose

Integration with docker-compose and podman-compose

Podman integrates with the **docker-compose** utility to orchestrate the execution of complex container environments. Alternatively, users can choose the native **podman-compose** utility which also implements the Compose spec. **This integration is not yet supported on RHEL8.**

```
# Run a compose stack with docker-compose
$ sudo dnf install docker-compose podman-docker
$ sudo systemctl enable --now podman.socket
$ docker-compose up

# Run a compose stack with podman-compose
$ sudo dnf install -y podman-compose
$ podman-compose up
```



Generating and running Kubernetes manifests

`podman generate kube, podman play kube`

The suggested and supported approach to run complex container stacks is the usage of Kubernetes manifest files. Podman can generate and play **Pod**, **Service** and **PersistentVolume** manifests, allowing the execution of complex stacks and easier porting to Kubernetes.

```
# Generate and play a manifest from a running multi-container pod
$ podman generate kube wordpress-pod -f wordpress-single-pod.yaml
$ podman play kube wordpress-single-pod.yaml

# Generate a manifest from multiple pods and play it over a custom network
$ podman generate kube wordpress-pod mysql-pod -f wordpress-multi-pod.yaml
$ podman play kube --network custom-net wordpress-multi-pod.yaml
```

Interactive Labs



[Topics](#) [Products](#) [Developer Sandbox](#) [Build](#) [Tools](#) [Events](#) [Learn](#) [Partner](#)

[All learning](#) > [OpenShift learning](#)

> Lesson

Get started with containers

20 minutes |

Learn the basics of a typical container architecture. This will cover container images, registries, hosts, and orchestration.

[Start →](#)

> Lesson

Manage your containers

35 minutes | Intermediate

Understand how container images are built, tagged, organized, and leveraged to deliver software in a range of use cases.

[Start →](#)

> Lesson

Understand container registries

35 minutes | Intermediate

Further your understanding by learning what container registries are for and how they work.

[Start →](#)

Further Readings

Community Documentation

- Podman official documentation
<https://docs.podman.io/en/latest/>
- Podman GitHub repository
<https://github.com/containers/podman>
- Podman Blog
<https://podman.io/blogs/>

Further Readings

Podman for DevOps

Containerization reimaged with Podman and its companion tools

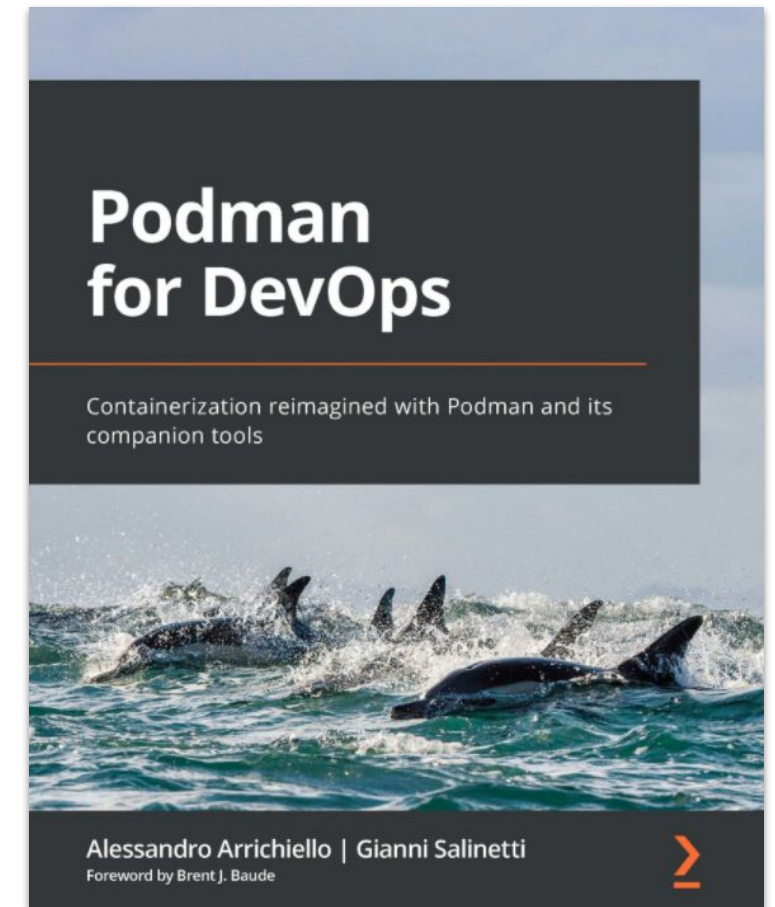
Authors: Alessandro Arrichiello, Gianni Salinetti

Publisher: Packt Publishing

ISBN-13: 978-1803248233

Available on Amazon and Packtpub.com

<https://github.com/PacktPublishing/Podman-for-DevOps>



Connettiti con Red Hat!



Scansiona il QR code per restare aggiornato con le ultime novità!

Allo Stand c'è un gadget che ti aspetta, e potrai anche parlare con i nostri esperti.

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.



linkedin.com/company/red-hat



youtube.com/user/RedHatVideos



facebook.com/redhatinc



twitter.com/RedHat