What is Podman?

- Podman is a daemonless, open source, Linux native tool designed to make it easy to find, run, build, share and deploy applications using Open Containers Initiative (OCI) Containers and Container Images.
- Podman provides a command line interface (CLI) familiar to anyone who has used the Docker Container Engine. Most users can simply alias Docker to Podman without any problems.
- Similar to other common Container Engines (Docker, CRI-O, containerd), Podman relies on an OCI compliant Container Runtime (runc, crun, runv, etc) to interface with the operating system and create the running containers.
- This makes the running containers created by Podman nearly indistinguishable from those created by any other common container engine.
- Containers under the control of Podman can either be run by root or by a non-privileged user.
- Podman manages the entire container ecosystem which includes pods, containers, container images, and container volumes using the libpod library.
- Podman specializes in all of the commands and functions that help you to maintain and modify OCI container images, such as pulling and tagging.
- It allows you to create, run, and maintain those containers and container images in a production environment.

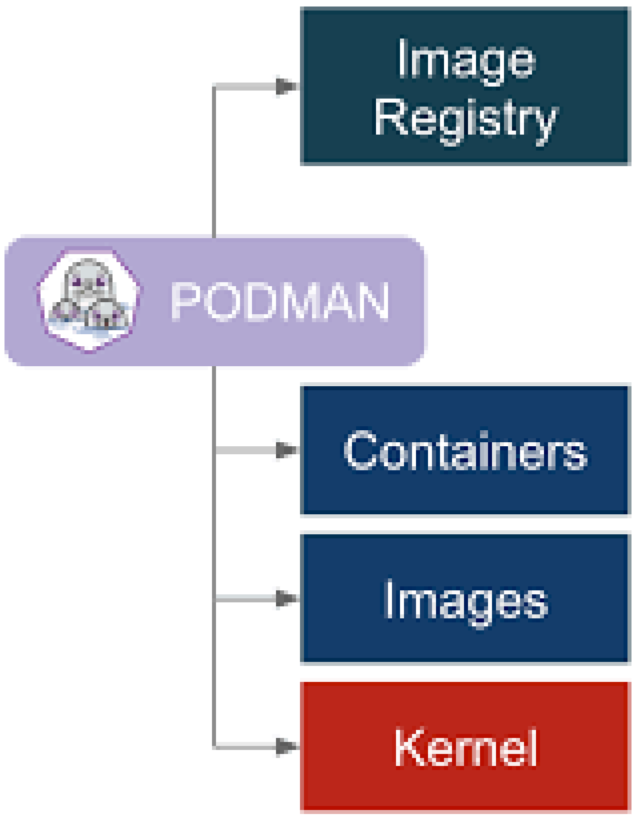
Podman vs. Docker

Feature	Docker	Podman
Architecture	Uses a single daemon (dockerd) to manage containers.	Daemonless; containers run as independent processes.
Process Management	Containers depend on the daemon process, which can be a single point of failure.	Containers run as child processes of the user's shell, reducing reliance on a single entity.
Rootless Operation	Supports rootless mode with additional configuration.	Designed to run rootless by default, offering better security.
SElinux Support	Requires manual configuration for SELinux integration.	Native SELinux support out of the box.

Sumeha Jakate

Container Images	Pulls container images from Docker Hub and supports the OCI format.	Fully compatible with Docker images and uses the same OCI image format.
Commands	Docker CLI is specific to Docker.	Supports Docker-compatible commands for easier transition (e.g., podman run).
Networking	Uses a default bridge network for containers.	Relies on CNI (Container Network Interface) plugins for networking.
Orchestration	Integrates with Docker Compose for multi-container setups.	Uses podman-compose, though less mature compared to Docker Compose.
Volumes	Provides robust volume management.	Supports volumes with slight differences in implementation.
System Integration	Can be manually integrated with systemd.	Offers built-in support for generating systemd unit files.
Installation	Requires installation of the Docker daemon and client.	Lightweight installation without needing a daemon.
Use Cases	Ideal for environments requiring orchestration and an extensive ecosystem.	Suitable for secure, rootless, and lightweight environments

Podman Architecture



- **Image Registry**: Podman pulls container images from registries like Docker Hub or other OCI-compliant registries.
- **Containers**: It runs and manages containers directly without the need for a daemon.
- **Images**: Podman uses OCI-compliant images to create containers.
- **Kernel**: Containers are built on the host's kernel, leveraging Linux kernel features like namespaces and cgroups for isolation and resource management.

Limitations of Podman

- **Lack of Mature Orchestration Tools**

  Podman does not have a built-in orchestration tool equivalent to Docker Swarm or native Kubernetes integration.

  Orchestration can be achieved using tools like Kubernetes or podman-compose, but they are less mature compared to Docker Compose.

- **Compatibility with Docker Compose**

  While podman-compose exists, it is not as feature-complete or stable as Docker Compose.

  Transitioning complex Docker Compose setups to Podman may require extra effort.

- **Networking Difference**

  Podman uses CNI (Container Network Interface) for networking, which may require manual configuration, especially for advanced setups.

  It does not support Docker's default bridge networking behavior.

- **Windows Support**

  Podman’s primary support is for Linux, with limited functionality on macOS and Windows through virtualization (e.g., using WSL2 or VMs).

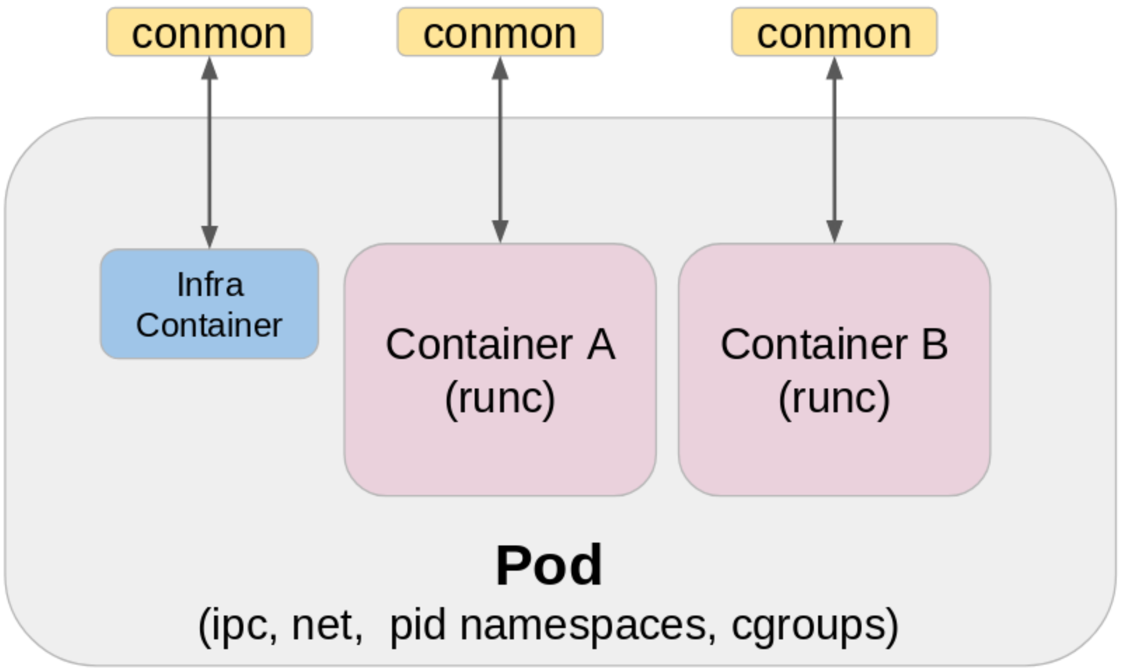
  Docker, on the other hand, offers more seamless support for Windows environments.

- **Smaller Community Support**

  Podman’s community is smaller compared to Docker, which might make finding resources, tutorials, or troubleshooting help more challenging.

Pods in Podman

Pods are groups of containers that run together and share the same resources, similar to Kubernetes pods. Podman manages these pods via a simple command-line interface (CLI) and the libpod library, which provides application programming interfaces (APIs) for managing containers, pods, container images, and volumes. Podman's CLI creates and supports Open Container Initiative (OCI) containers, which are designed to meet industry standards for container runtimes and formats. More advanced building capabilities are available in the related project, Buildah.



Each pod is composed of 1 infra container and any number of regular containers. The infra container keeps the pod running and maintains user namespaces, which isolate containers from the host. The other containers each have a monitor to keep track of their processes and look out for dead containers—nonfunctioning containers that can't be taken out of the environment because some of their resources are still being used.

Sumeha Jakate

Why Podman?

Podman changed the container landscape by offering the same high-performance capabilities as leading container engines, but with the flexibility, accessibility, and security features that many development teams are seeking. Podman can help you:

- Manage container images and the full container lifecycle, including running, networking, checkpointing, and removing containers.
- Run and isolate resources for rootless containers and pods.
- Support OCI and Docker images as well as a Docker-compatible CLI.
- Create a daemonless environment to improve security and reduce idle resource consumption.
- Deploy a REST API to support Podman's advanced functionality.
- Implement checkpoint/restore functionality for Linux containers with Checkpoint/Restore in Userspace (CRIU). CRIU can freeze a running container and save its memory contents and state to disk so that containerized workloads can be restarted faster.
- Automatically update containers. Podman detects if an updated container fails to start and automatically rolls back to the last working version. This provides new levels of reliability for applications.