Containers in HPC – Podman

UManitoba Spring 2026
High-Performance Computing And Cloud
Workshop

Stefano Ansaloni

University of Manitoba

May 21, 2026

Stefano Ansaloni

Cloud Computing Specialist at University of Manitoba
(part of the Advanced Research Computing team)

Software Developer and DevOps Specialist since 2017

Linux User/Admin since 2005

University of Manitoba

What is a container

From Wikipedia (Containerization):

> *Containerization is operating-system–level virtualization or application-level virtualization so that softwares can run in isolated user spaces called "containers" in any cloud or non-cloud environment, regardless of type or vendor.*

Properties of containers

- Each container is a fully functional and portable computing environment surrounding the application and keeping it independent of other environments running in parallel
- Each container simulates a different software application and runs isolated processes (including configurations, libraries, and dependencies)
- Multiple containers share a common operating system kernel (operative system)

Wikipedia [Containerization]

University of Manitoba

Terminology

Image	Archive (or number of archives – i.e. "layers") of a filesystem tree along with metadata
Containerfile	Recipe for building an image, including OS and software within the image (e.g. *Dockerfile*)
Container	A running instance of an image (can be a computing process, or a service daemon)
Container Runtime	Lower level component responsible for reading the image and communicating with the host kernel to start containerized processes (e.g. *runc*, *crun*)
OCI (Open Container Initiative)	Open governance structure that creates open industry standards for container formats and runtimes
Registry	An online storage area for images (e.g. *DockerHub*, *Quay.io*)

Focusing On Podman

What is Podman

From Wikipedia (Podman):

> *Podman (pod manager) is an open source OCI-compliant container management tool from Red Hat used for handling containers, images, volumes, and pods; offering APIs for the lifecycle management of those components (the API is identical to the Docker API).*

Why not Docker

Podman aims to provide a more secure and lightweight alternative to Docker:

- Daemonless ⇒ Don't rely on a process with root privileges to run containers
- Rootless containers ⇒ Run containers as regular users, without interacting with a root-owned daemon
- User namespaces ⇒ Careful use of kernel capabilities

Red Hat [What is Podman]
Better Stack [Exploring Podman]

University of Manitoba

Compatibility with Docker

For most use cases, Podman can be used as a “drop-in” replacement for Docker:

- Podman CLI syntax is almost the same as Docker’s one
- Podman can use the same images as Docker
- Podman can use the same registries as Docker

Podman on HPC – Disclaimer

Podman is considered an advanced tool to be used only by experienced users when their workloads cannot be run using standard HPC programs/modules, or through Singularity/Apptainer (i.e. Podman should be the very last resort).

In any case, Podman must **NOT** be used:

- on login nodes
- to execute long-running services (e.g. daemons, databases, ...)

Podman on HPC

Podman is available as a module on national clusters and UManitoba HPC cluster (Grex).

HPC System	Command	Current version
National Clusters	`module load StdEnv/2023 podman`	5.7.1
Grex	`module load podman`	5.8.2

Podman on Grex

When on Grex, it is important to use the local version of Podman:

- local proxy cache for registries
- better default configuration
- newer version

Basic commands

Print version	`podman version`
Pull image	`podman pull <REGISTRY>/<NAME>:<TAG>`
Delete image	`podman image rm <IMG>`
List images	`podman image ls`
Create and start a container	`podman run [OPTS] <IMG> [CMD [ARGS]]`
Execute command inside a running container	`podman exec [OPTS] <CNT> [CMD [ARGS]]`
Stop a running container	`podman stop <CNT>`
Start a stopped container	`podman start <CNT>`
Delete a container	`podman rm <CNT>`
List containers	`podman ps [-a]`

Getting an image

From a registry

The "`pull`" subcommand can be used to download an image from an online registry (e.g. *DockerHub*) into a local (temporary) registry, that is available only on the node where the *pull* has been performed and will be deleted when the job ends.

```
podman pull docker.io/alpine
```

Getting an image

Building your own

If it becomes necessary to build a custom image, it can be accomplished with the "`build`" subcommand.

By default the built image will be saved into a local (temporary) registry, that is available only on the node where the *build* has been performed and will be deleted when the job ends.

To overcome this problem, the "`--tag`"/"`-t`" option can be specified using "`docker-archive:`" as the *tag* prefix.

```
podman build -t docker-archive:/path/to/custom.image \
             -f /path/to/Containerfile
```

Binding host directories

To let containers access data on the host (e.g. the user home directory), the "`--volume`" / "`-v`" option can be specified (multiple times) using the format "`HOST_DIR:CNT_DIR`".

```
podman run -v $HOME:$HOME docker.io/alpine ls $HOME
```

When binding directories, the container will be able to read and write the content of those directories (if the user has the correct filesystem permissions).

Setting environment variables

The option "--env" / "-e" can be specified (multiple times) to set environment variables inside the container.

```
podman run -e V1="hi" -e V2="$USER" docker.io/alpine sh -c 'echo $V1 $V2'
```

If the value is not specified, Podman will map the host variable value into the container.

```
podman run -e USER docker.io/alpine sh -c 'echo $USER'
```

Demo

Running a GROMACS benchmark with Podman

Running a GROMACS benchmark with Podman

Specifications

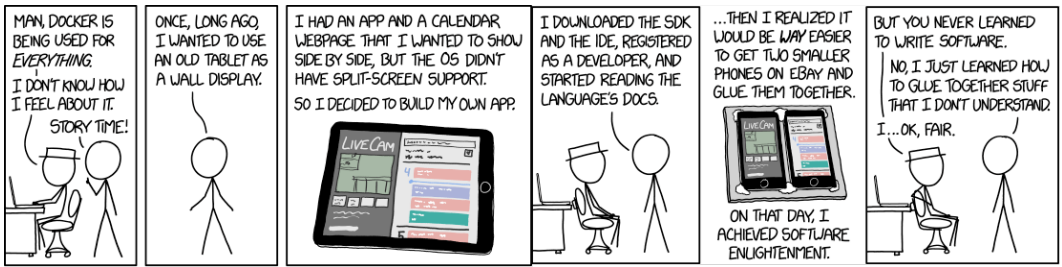
Container Engine	⇒	Podman (on Grex)
Image	⇒	nvcr.io/hpc/gromacs:2023.2 (from the nVidia NGC Catalog)
Software	⇒	GROMACS
Benchmark	⇒	STMV

University of Manitoba

Is Podman a silver bullet?

- Always check if the same software is already provided via modules-based HPC software stack
- It requires *well-built* images
  - Making or finding a suitable image is a bit of work
  - Bleeding-edge versions of programs could be poorly maintained/tested (including their images)
- Useful to encapsulate software and sometimes data to reduce number of files (e.g. python or conda based programs)

University of Manitoba



XKCD [1988]

University of Manitoba

Questions?

Thank you