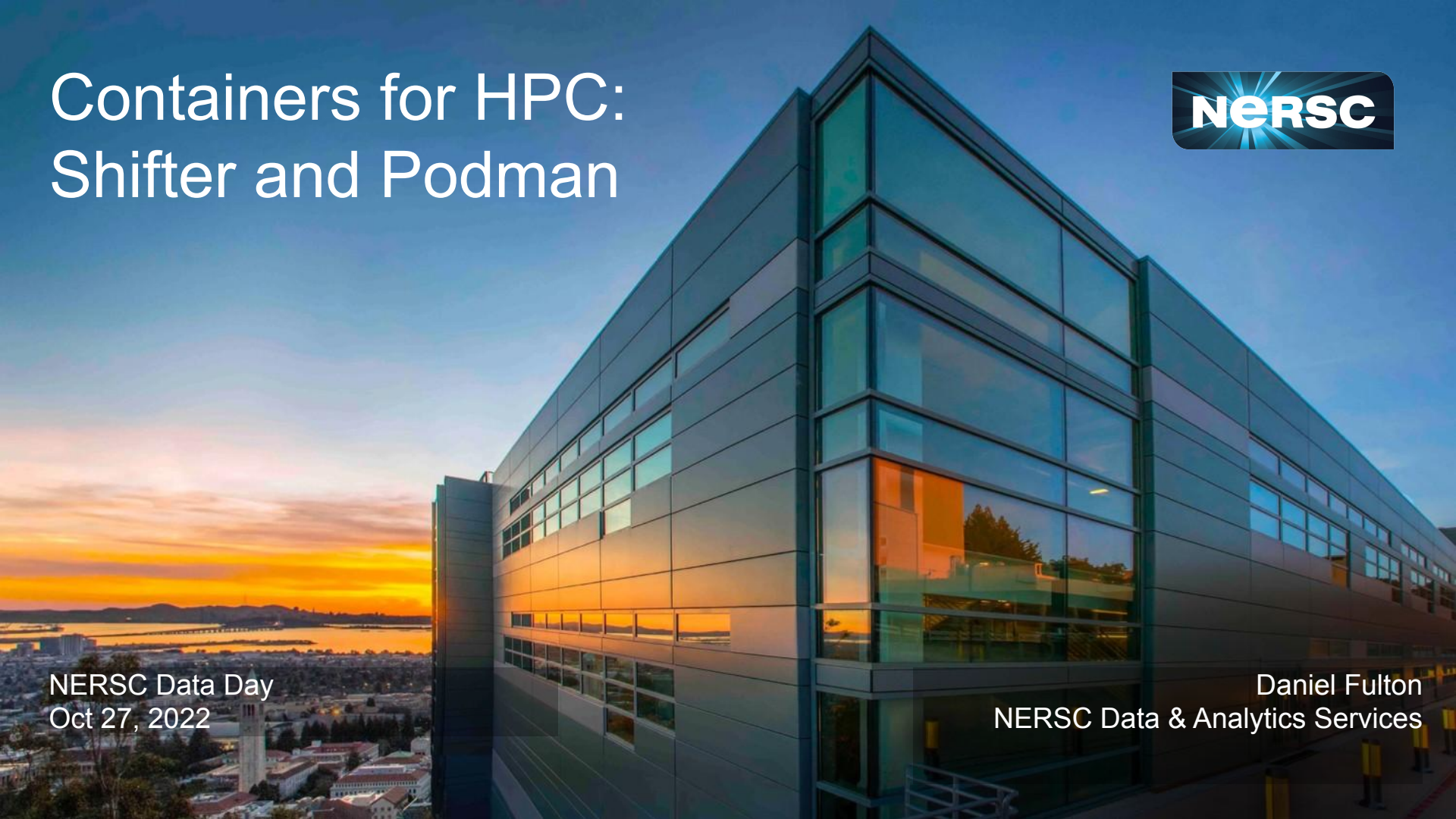
Containers for HPC: Shifter and Podman

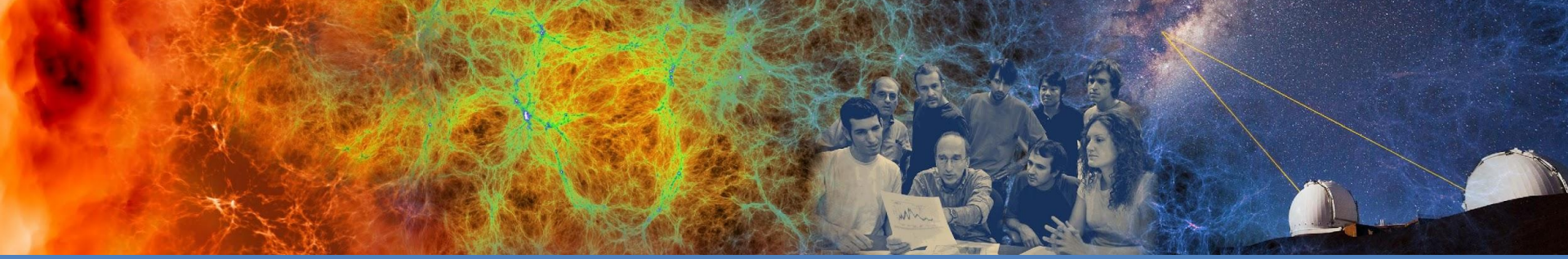


NERSC Data Day
Oct 27, 2022

Daniel Fulton
NERSC Data & Analytics Services

Outline of This Talk

- A Very Brief Introduction to Containers
- Today: Using Shifter at NERSC
- Tomorrow: Using Podman at NERSC

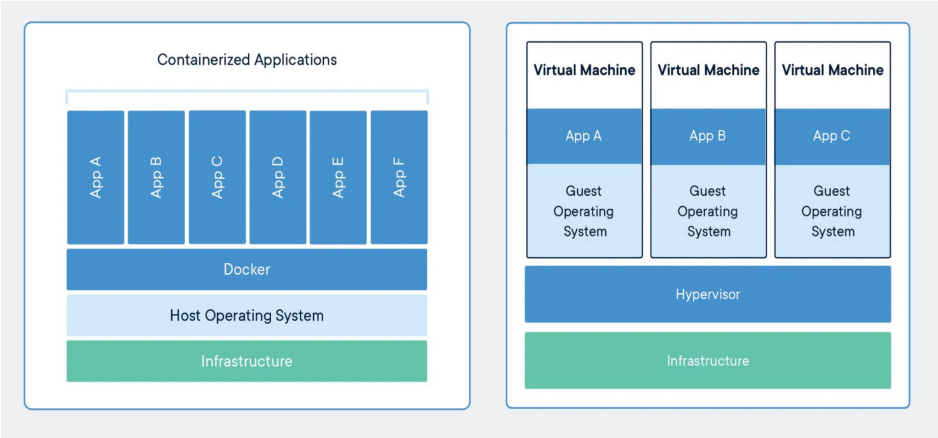


A Very Brief Introduction To Containers

What is a container?

- A container is similar in purpose to a virtual machine (VM), providing encapsulation for a software application and it's runtime environment.
- Implementation differs. Containers use the host linux kernel instead of virtualizing hardware, so they are lightweight compared to VMs.
- Linux Containers rely on kernel features, and are inherently Linux based.

Container

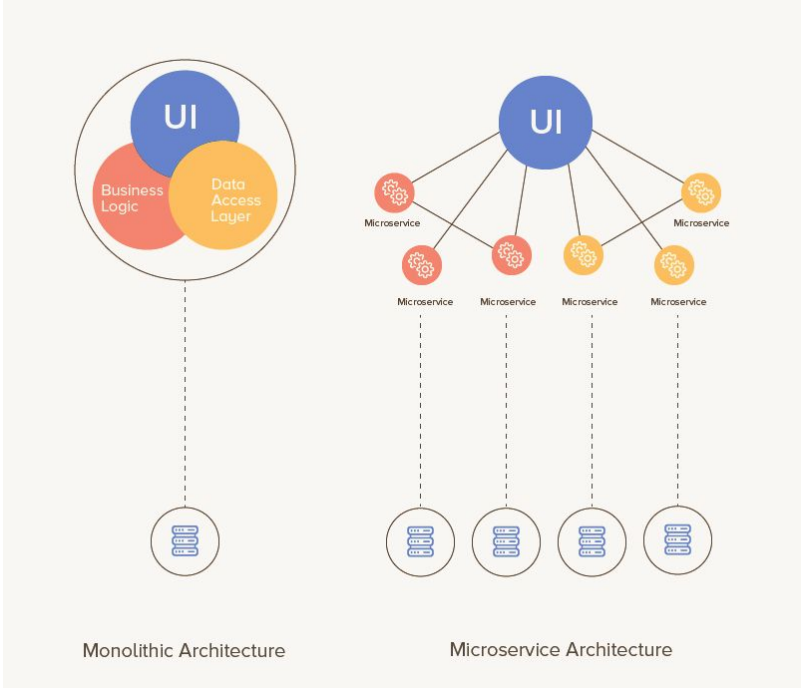


Virtual machine

Image from https://www.docker.com/resources/what-container/

Why are containers popular?

- Smaller footprint than a VM with nearly all of the encapsulation benefits, including:
  - Portability
  - Scalability
  - Reproducibility
- Switch from Imperative to Declarative paradigm improves reproducibility.
- Building block of modern scaleable web applications, e.g. “microservice architecture”.



What do containers bring to HPC?

Portability, Reproducibility, Scalability (but our use cases differ slightly):

- Build once, use by many
- Isolate from changes in HPC software env.
- Save simulation/analysis software runtime for reproducible science.
- Avoid metadata contention (e.g. Python) on a shared filesystem.
- Move to a different supercomputer!
- Scientists also like web applications (data portals, workflow management)

Interlude: Container Vernacular

- **Image**- An archive of an application and it's runtime environment.
- **Container**- Running instance of an image, w/ ephemeral filesystem on top.
- **Container runtime**- Software responsible for launching and running a container instance from an image.
- **Container engine-** Higher level container framework, which typically includes an image builder and container runtime.
- **Dockerfile/Containerfile**- Human readable file which specifies instructions for a container engine to build an image.
- **Image Registry**- Network/cloud accessible storage repository for images. May be public or private.
- **Volume mount/Bind-mount-** A way to mount persistent files or directories into a container at run time.

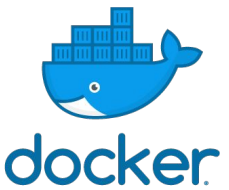
Interlude Continued: Household Names

- **Docker, Podman-** Popular container engines.
- **Shifter, Singularity-** HPC specific container engines.
- **Dockerhub, quay.io-** Popular public container registries.
- **Harbor-** An open source image registry implementation.
- **Docker Desktop, Rancher Desktop-** MacOS/Windows clients to manage a Linux VM which has been optimized to run a container engine.
- **Open Container Initiative (OCI)-** Open standards body focused on governance of Linux containers.
- **Kubernetes (K8s)-** An open source standard for *orchestration* of container deployment, scaling, and management.
- **SUSE Rancher, RedHat OpenShift, Amazon EKS, Google GKE, Azure Kubernetes Services, usernetes, k3s, minikube-** Implementations of K8s.
- **Cloud Native Computing Foundation (CNCF)-** Governing body for K8s.

Sample Container Workflow

Build

```
> vim ./Dockerfile
> docker build -t me/myimage:latest .
```



Ship

```
> docker push me/myimage:latest
```

Run

```
> docker pull me/myimage:latest
> docker run me/myimage:latest
```

Sample HPC Container Workflow?

Build

```
> vim ./Dockerfile
> docker build -t me/myimage:latest .
```

Ship

```
> docker push me/myimage:latest
```

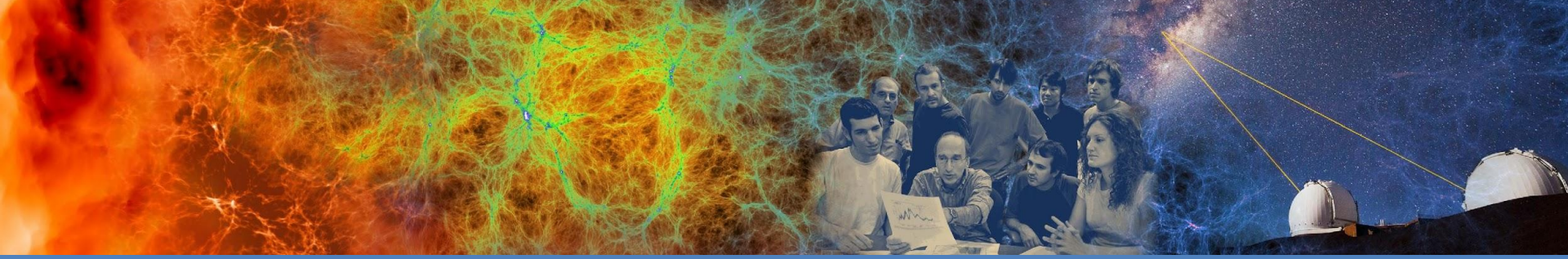
Run

```
> docker pull me/myimage:latest
> salloc -C cpu
> srun --ntasks-per-node=$SLURM_TASKS_PER_NODE
docker run me/myimage:latest
```

WON'T WORK

Considerations for Containers on HPC

- HPC applications may be sensitive to filesystem performance.
- HPC applications may be very communication intensive.
- A multiuser HPC system is not a trusted environment.
- How can optimized HPC libraries be easily included in a containerized HPC application?
- How does container launch interact with the batch scheduler?



Today: Using Shifter at NERSC

Shifter at NERSC

- Shifter has been the container engine at NERSC since it was introduced in 2015.
- Shifter is increasingly popular, with 700+ unique users in the first half of 2022.
- Shifter addresses the problems of running containers on HPC.

SHIFTER

Shifter addresses the problems…

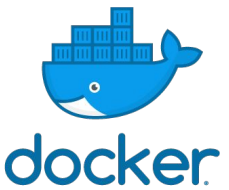
Problem	Shifter…
Sensitivity to filesystem performance.	squashes layered image into a single-layer read-only image. `> shifterimg pull me/myimage:latest`
Communication intensive performance.	opts out of virtualized networking and passes through high-performance HPC network.
Security in multiuser environment.	requires containers to run as non-root.
Including optimized HPC libraries.	optionally, can hook system libraries into a container at runtime. `> shifter --module=gpu --image=me/myimage:latest`

Batch scheduler interaction.	has some configuration options passed via Slurm
	`> salloc -C gpu --image=me/myimage:latest`
	`> srun shifter`

Sample Shifter Workflow

Build

```
> vim ./Dockerfile
> docker build -t me/myimage:latest .
```



Ship

```
> docker push me/myimage:latest
```

Run

```
> shifterimg pull me/myimage:latest
> salloc -C cpu --image=me/myimage:latest
> srun --ntasks-per-node=$SLURM_TASKS_PER_NODE
shifter
```

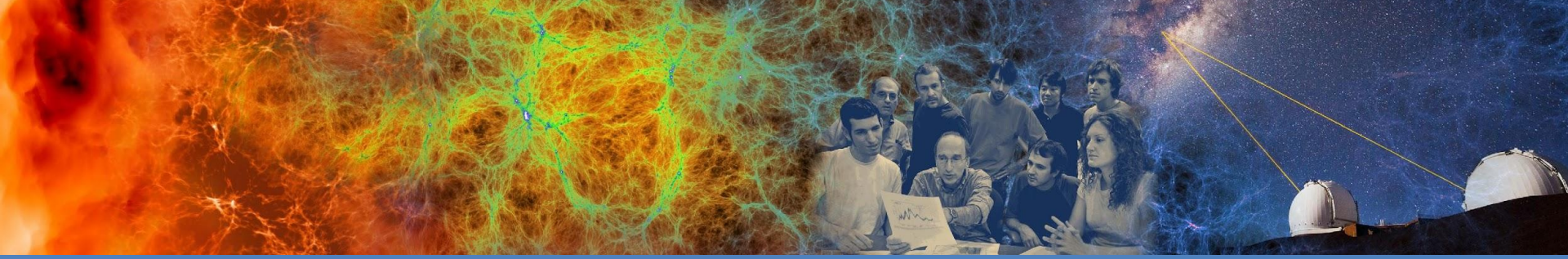
Learn More About Shifter

- Learning to use Docker on your laptop is a good place to start.
- Check out the excellent Shifter training talk by Laurie Stephey given during Sep 2022 New User Training.
- Check out our Shifter docs and beginner tutorial to learn more.
- **`shifter --help`**
- If you get stuck, please contact us at help.nersc.gov so we can help!

Why not stick with Shifter?

Shifter has several shortcomings:

- No builder included, users can't build at NERSC and it is harder to include optimized HPC libraries.
- Requirement to run as non-root user disallows many off-the-shelf containers and complicates container design.
- Shifter is maintained "in-house" at NERSC. Difficult to provide manpower for future development. Difficult for users to learn another unique tool.



Tomorrow: Using Podman at NERSC

Podman addresses Shifter's weaknesses

- Podman (Pod manager) is an Open Container Initiative compliant container framework under active development by Red Hat, Inc.
- Free, open source, and widely used by an active community.
- Provides full-features *rootless* containers by mapping root inside container to user pid space, providing a secure multiuser engine.
- Provides an image builder.
- Shares CLI syntax with Docker.
- *Can HPC performance be achieved via additional configuration?*

podman

Bringing Shifter Performance to Podman

- Enabled squashed images using a wrapper before performing the overlay mount. This wrapper also handles cleanup for the squash mount after the container is removed.
- Podman allows using pass-through host networking.
- Podman allows specifying custom hooks (e.g. mpich and gpu libraries)
- Experimentation to determine an efficient way to launch multiple podman instances with `srun`.

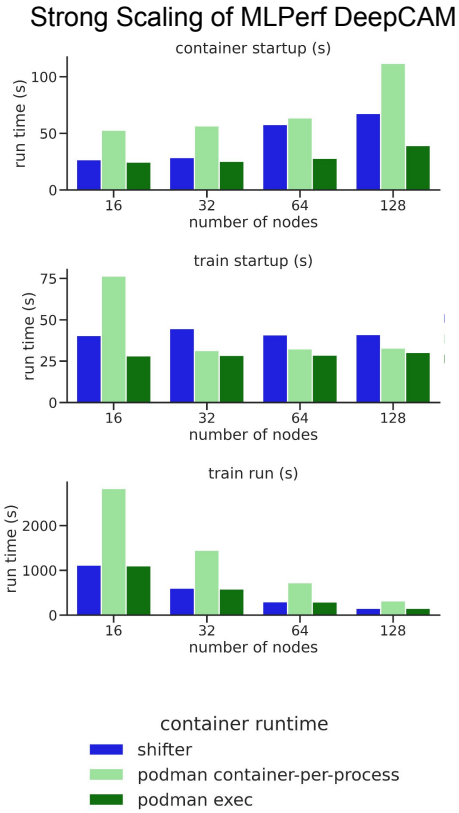
To simplify this extensive configuration, NERSC has created a `podman-hpc` wrapper to extend `podman` functionality, while simplifying HPC specific setup by the user.

Performance Benchmarks

- Ran four different benchmarks to evaluate **`podman-hpc`**
  - Pynamic (CPU, Python, metadata-heavy)
  - AstroPy+mpi4py import (CPU, Python, metadata-heavy)
  - EXAALT (GPU, Kokkos, traditional simulation)
  - DeepCAM (GPU, Python, part of MLPerf suite)
- Compared bare-metal, Shifter, and two Podman configurations
- Ran up to 256 nodes (system, not Podman, limitations)

Podman can perform comparatively or even better than Shifter when configured appropriately.

For details see upcoming CANOPIE-HPC paper "*Scaling Podman on Perlmutter: Embracing a community-supported container ecosystem*" Laurie Stephey, et al.



Sample podman-hpc Workflow

Build

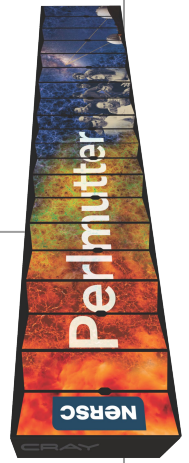
```
> vim ./Dockerfile
> podman-hpc build -t me/myimage:latest .
```

~~Ship~~

```
~~> docker push me/myimage:latest~~
```

Run

```
> podman-hpc mig me/myimage:latest
> salloc -C cpu
> srun --ntasks-per-node=$SLURM_TASKS_PER_NODE
podman-hpc run-shared me/myimage:latest
```



Building with Podman-HPC

```
# build from a Dockerfile
> vim ./Dockerfile
> podman-hpc build -t myimage:latest .
# OR pull an image from elsewhere
> podman-hpc pull myimage:latest
```

One 3TB NVMe drive per Perlmutter login node at **`/images`** to support rootless podman image builds. Separate login nodes have separate container stores.

```
# migrate the image to squashed, read-only
> podman-hpc mig myimage:latest
```

Migrating an image creates a squashed, read-only copy in a separately configured storage location.

Shipping with Podman-HPC

- With container builds enabled on Perlmutter, pushing images to a registry is optional, but a strongly recommended best practice.
- Registries
  - NERSC registry.nersc.gov - private, free with NERSC account
  - DockerHub- public, free OR private, paid
  - Quay.io- public, free OR private, paid

```
# retag an image and push it to NERSC registry
> podman-hpc login registry.nersc.gov
> podman-hpc tag myimage:latest registry.nersc.gov/das/myimage:1.0.0
> podman-hpc push registry.nersc.gov/das/myimage:1.0.0

# pull the same image down later
> podman-hpc pull registry.nersc.gov/das/myimage:1.0.0
```

Running with Podman-HPC

For brief container usage on logins, call **`run`** normally:

```
> podman-hpc run myimage:latest
```

Within batch allocations use **`run-shared`** to launch one container per node, and one process per thread inside the container:

```
> salloc -C cpu
> srun --ntasks-per-node=$SLURM_TASKS_PER_NODE podman-hpc
run-shared myimage:latest
```

Add **`--gpu`** or **`--mpi`** flags to hook optimized system libraries from Perlmutter into your container at runtime:

```
> salloc -C cpu
> srun --ntasks-per-node=$SLURM_TASKS_PER_NODE podman-hpc
run-shared --mpi --gpu myimage:latest
```

Summary and Future Plans

- Shifter currently provides good container performance on Cori and Perlmutter, however…
- Podman has demonstrated comparable performance, and will provide many additional benefits:
  - Community supported
  - Standardized interface
  - Full end-to-end container engine
- Working **`podman-hpc`** wrapper coming soon. Syntax improvements still underway and subject to change.
- Strong collaboration with RedHat and will upstream whatever makes sense.
- Shifter and Podman-HPC will coexist while users transition.

Thank you!