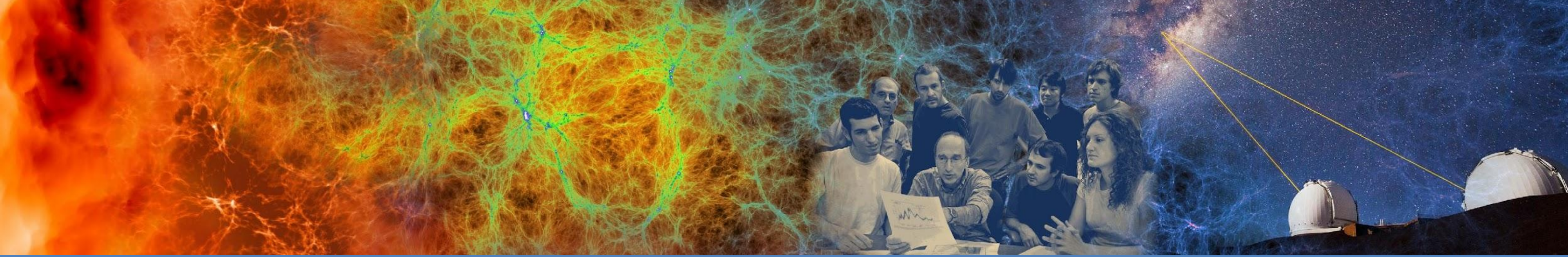
Containers for HPC Session 1

Slides: https://bit.ly/20250313Container
Q&A Doc & Survey

NERSC Containers Training
13 March 2025

NERSC Staff



Today’s Container Training

What today is:

A general training about container basics with hands on exercises:

- Containerfiles, building containers, pulling containers, using containers interactively, submitting jobs using containers

An introduction to more advanced ways to use containers:

- Containers as Jupyter kernels, Spin

What today is not:

- A training on general HPC practices
  - I link to job type stuff for those of you who are not familiar
- A deep dive into podman-hpc or Shifter
  - We will focus on using these, but not how they work
- A focus on any individual field’s container needs:
  - Submit a ticket and NERSC can help
- A way for you to get personalized help with your container needs
  - Submit a ticket and NERSC can help
- Detailed instructions on advanced topics
  - We will point to resources

Today’s Agenda

First session - 9 AM

- What is a container?
- Container runtimes
- Basic Terminology
- Registries
- Pulling containers
- Running containers
- Hands on exercises (20 min)
  - Pull a container
  - Run interactively
  - Run as a batch job

Second session

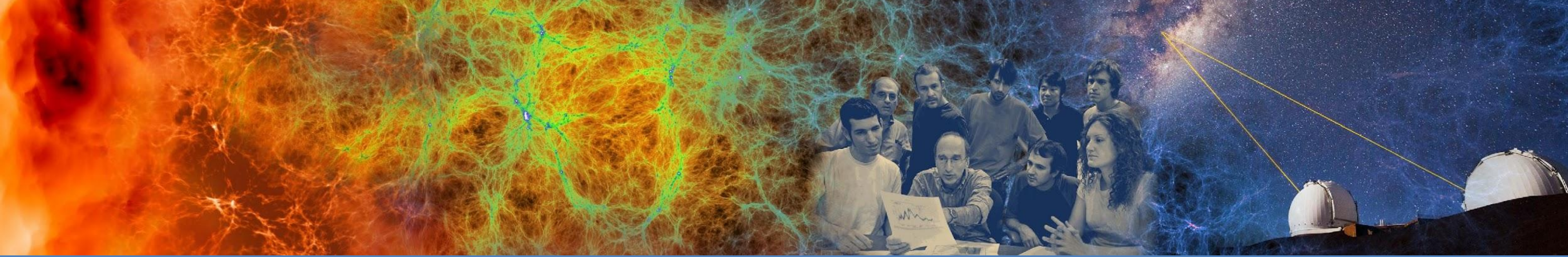
- Container workflows
- Building containers
- Container permissions
- MPI
- Hands on exercises (20 min)
  - Build an MPI container
  - Push to a registry
  - Run this

Third session

- Tips and Tricks
- Fixing issues
- Intro to Spin
- Containers as Jupyter kernels
- Wrap-up

How do I interact today?

- Ask questions in the google document
    - NERSC staff around to answer
- Breakout rooms are available for the exercises
    - Ask a question in the google document
    - NERSC staff members will either answer there or ask you to join a zoom
    - Feel free to share your screen in the breakout rooms!
- Fill out the questionnaire at the end



Introduction

What is a Container?

An answer:

An encapsulated software environment that runs using a separate linux kernel

What does this mean?

- You can package the software and data you think is important together
  - Reproducibility, portability, scalability, consistency
- This package runs using a container runtime
  - Various runtimes exist on/for different systems
- The package uses the host machine's linux kernel
  - This is much more efficient than every environment running independently

Clarifying Related Topics

Open Container Initiative (OCI): Standards body pushing for standardization across container engines

Virtual Machines (VMs): Emulates the entire computer and accesses hardware through a hypervisor

- More isolated from the host machine
- More overhead associated with the applications

Kubernetes: An open source container orchestrator standard

- Used to deploy, scale and manage containers
- Many implementations: k8s, k3s, OpenShift, Rancher, Swarm, …
- Governed by Cloud Native Computing Foundation (CNCF)

Slurm: A job scheduler used on HPC systems

- Allocate resources for a particular job which may contain a container

Why use Containers?

In General:

- Personalized functionality that is portable and performant

Specific:

- Provide full environment for reproducibility
- Keep files intended for /home on faster storage
- Easily install and test library updates
- Include exact third party library versions & compilations with code versions
- Provide an isolated environment for individual tests and code development
- Avoid home directory usage for conda environment performance
- Containerfiles can be used as a lightweight method for sharing complicated compilation instructions
- Allow for consistent libraries between multiple development sites
- Share environments between users for clean and rapid development
- Easily deploy applications onto multiple disparate compute resources
- Allow for simple testing across a variety of environments
- Utilize cloud-based resources for testing
- Functionality test across multiple distros before code publication

Containers for HPC

HPC Applications:

- Are often sensitive to filesystem performance
- May be communication intensive
- Often use system tuned libraries for peak performance
- Are run on shared (untrusted) systems
- Typically use batch schedulers

NERSC container engines and tools are built to do this well.

Changes from standard container engines for these purposes are denoted with an HPC mark

Containers at NERSC

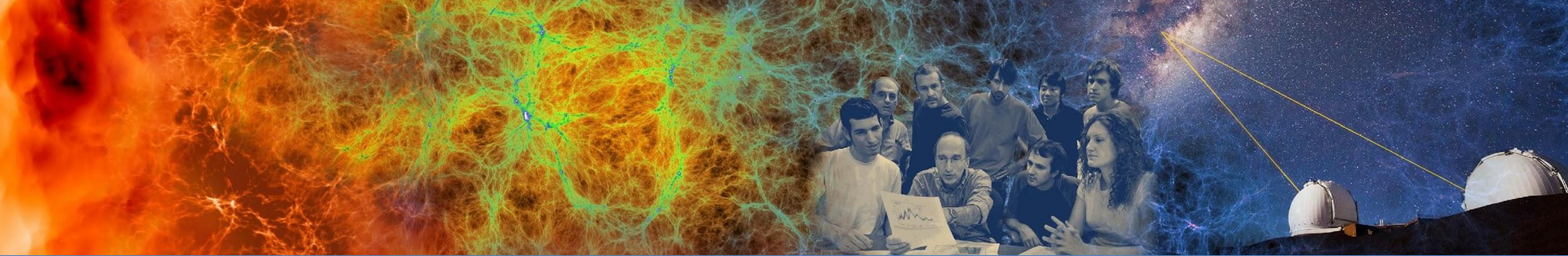
podman-hpc$^{\text{HPC}}$ is a NERSC built wrapper for podman

- HPC additions to a community supported container engine
- NERSC is building the HPC additions
- *Able to build images as a regular user*

Shifter$^{\text{HPC}}$ is a NERSC built container engine

- Built by NERSC to address HPC needs
- Popular at NERSC but not widely adopted elsewhere
- *Requires images to be built elsewhere*

Others: Docker, Apptainer, Charliecloud, Singularity, containerd, runC



Container Basics

Plain English Containers Workflow

Get a container image

- We can build this ourselves, or download one

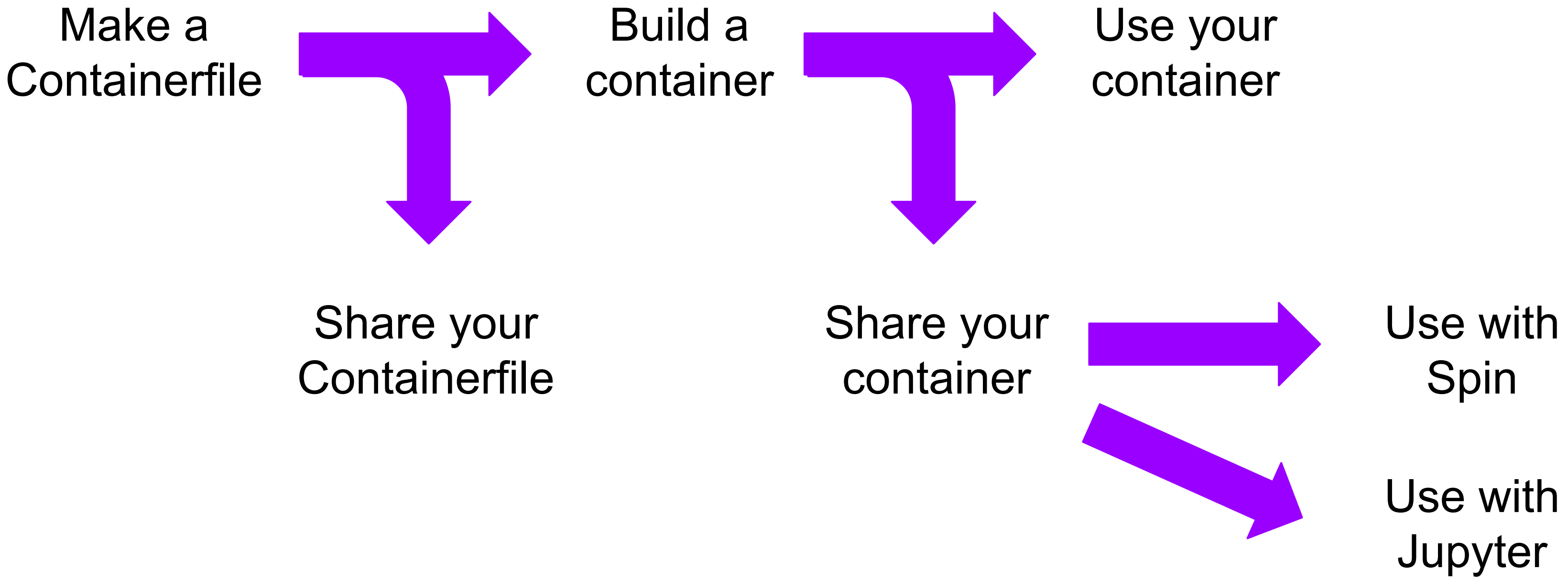
Run the container

- We use a program to run our container

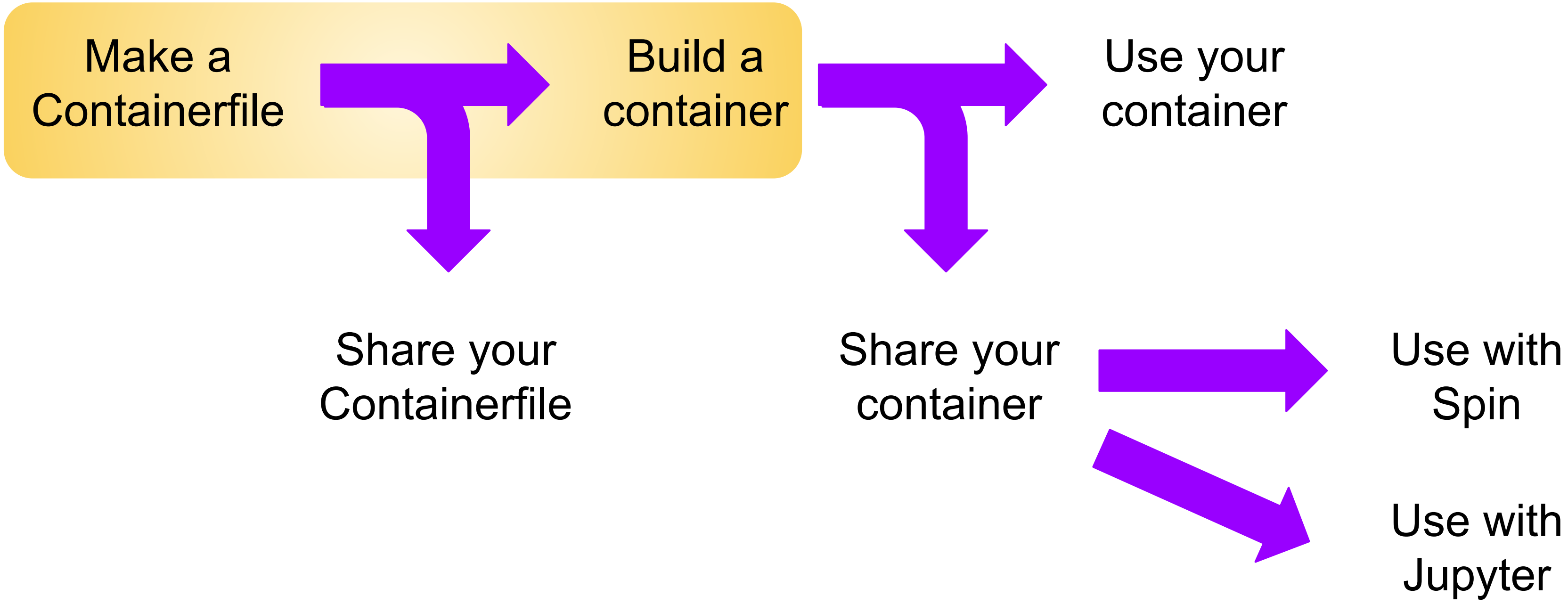
The next slides go through this in more detail with all of the container terminology.

Understand this basic version and you can pick up the terminology as you go!

Common Container Workflows



Common Container Workflows



Building a Container Image

Containerfile: a file that specifies how an image should be built

- Human readable
- Specify the OS and install libraries
- Get and compile files

Image Builder: a program that builds an image from a container file

- Either a separate program or a part of a container engine

Image: an archive of the environment, application, and data

- Binary file

Notes on Building a Container Image

Containerfile

Image Builder

Image

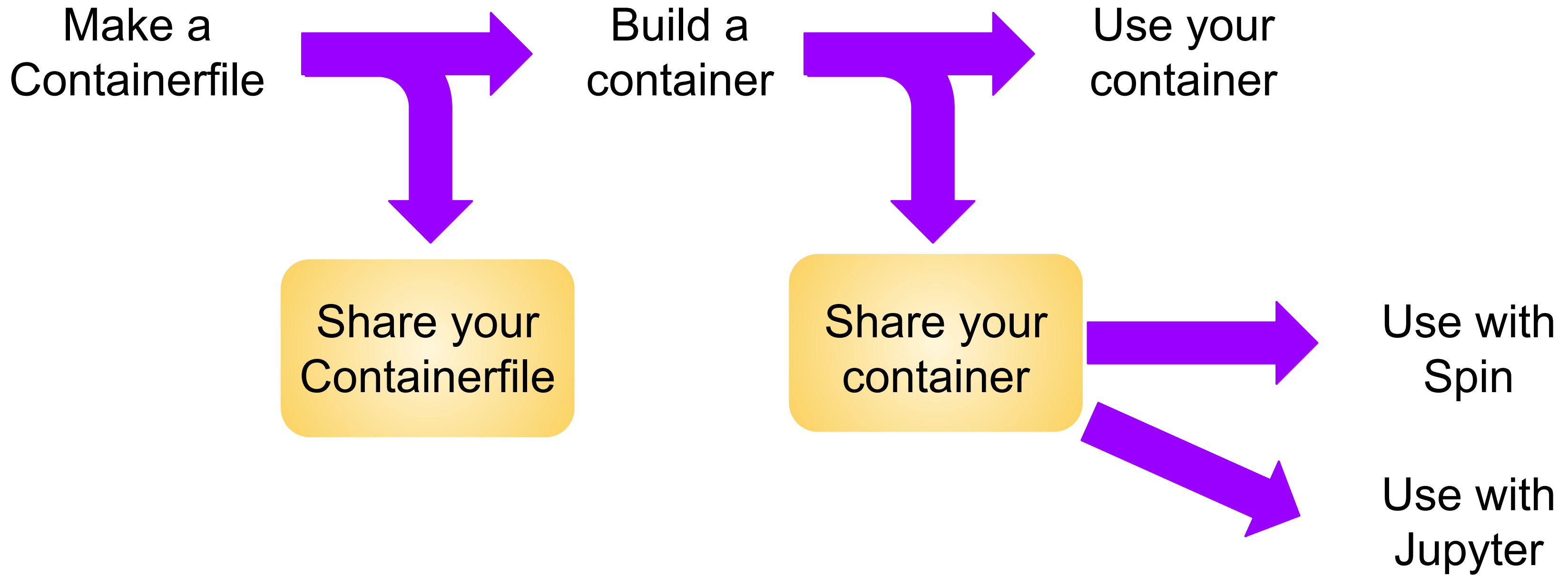
Containerfiles are commonly called **Dockerfiles**

- Docker was the first widely used container solution
- Docker nomenclature is still common even when Docker isn't being used

Images can be stored in an **image registry**

- Public or private
  - Dockerhub, quay.io, registry.nersc.gov
- Share your images or **pull** others already available

Common Container Workflows



Where/How to Share

Containerfiles: These are small (~kB)

- Slack, email, give/take
- Gitlab or Github for publishing
- You need to build this yourself

Container Images: These are large (~GB)

- Give/take
- Shared filesystems
- Registries

Container Registries

Harbor: registry.nersc.gov

- Internally run registry: NERSC can give you access
  - Put in a Service Now ticket
  - Projects can request a project space with a ticket from the PI
- Groups in harbor for all NERSC, NERSC staff and teams, research groups
- No access for non-NERSC users

Alternates:

- DockerHub: free public or paid private, widely used
  - https://hub.docker.com/u/nersc - some NERSC generated images
- Quay.io: free public or paid private, reasonable alternate

Other Specialized Registries

- NVIDIA has a well-stocked registry
  - Check out https://catalog.ngc.nvidia.com/containers

```
FROM nvcr.io/nvidia/nvhpc:24.5-devel-cuda_multi-ubuntu22.04
```

Tip: Older versions of CUDA are available if applications haven’t upgraded

- Perlmutter uses SLES
  - Check out https://registry.opensuse.org/cgi-bin/cooverview

```
FROM registry.opensuse.org/opensuse/bci/python:3.11
```

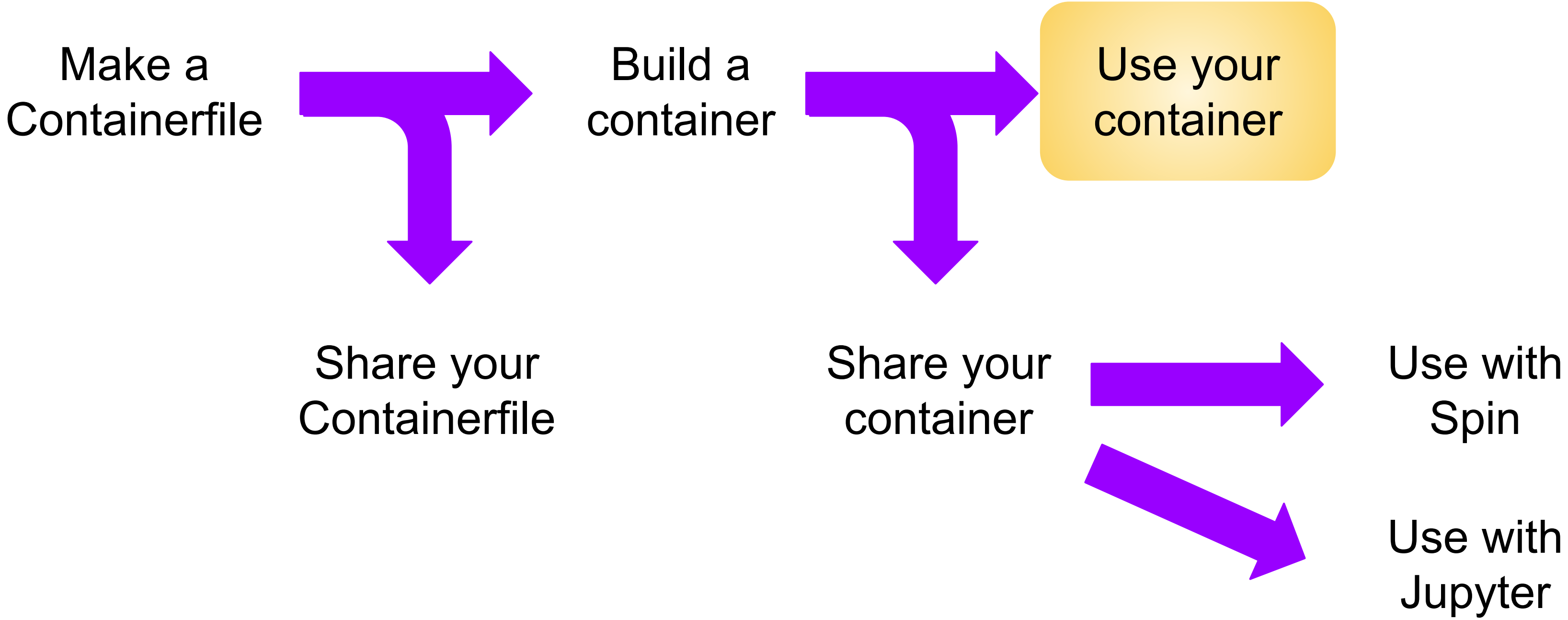
Tip: If you use a different OS, you may need to change your package manager commands

- Google can be an tool for finding images
  - Make sure you trust the source!

Tip: Many images link to their Containerfile - look through this to know what is there (and build yourself!)

- Github and Gitlab have nice tools for hosting

Common Container Workflows



Running a Container

Image (either built by you, shared with you, or pulled from a registry)

↓

Container engine: software used to launch containers

↓

- This is likely the command you use to launch a container

Container: an image that is running

- Application that you built and instructions for the run
- Likely includes ephemeral filesystem

Notes on Running a Container

Image

↓

Container engine

↓

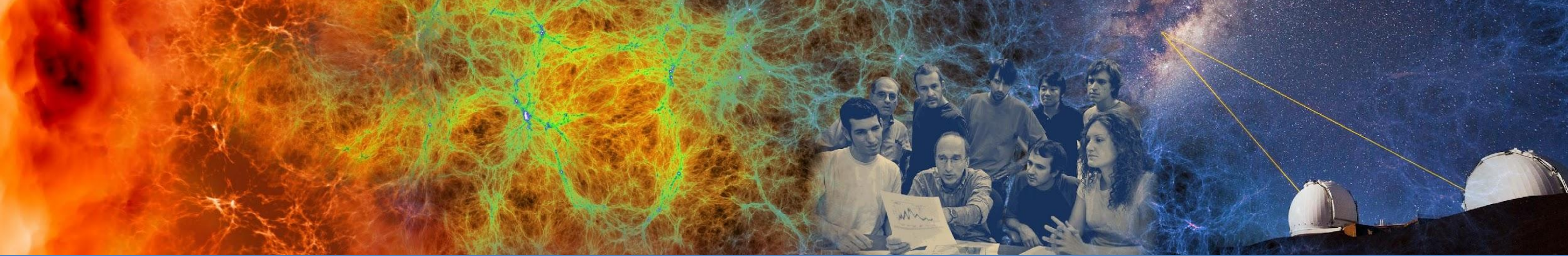
Container

Container Engines commonly have container runtimes and image builders

- Shifter, Apptainer, podman, Docker, CharlieCloud, …
- Different engines have different options available to regular users
- Often called *container back-ends*

Containers can have **volume mounts or bind mounts**

- Allows data from the host system to be available
- Allows optimized host system libraries to be used



Pulling and Running Basic Containers

Basic Podman-hpc^HPC Functionality

Pulling a container:

```
$ podman-hpc pull nvidia/cuda:11.0.3-base-ubuntu20.04
```

- Pull an image from a registry
- Username
- Container Name
- Tag/Version number
- Note this is automatically migrated to scratch^HPC

Note: you can pull any public container; you will need to login to pull private containers

View your container:

```
$ podman-hpc images
localhost/nolols             1.0          59551900ead8   3 minutes ago   80.4 MB
docker.io/library/ubuntu     24.04        3db8720ecbf5   8 days ago      80.4 MB
```

- View the images

Note: You only see your images with podman-hpc

Running Basic podman-hpc Containers

On the log-in node:

```
$ podman-hpc run --rm nolols:1.0
no lols here
```

- Use podman-hpc to run the container
- Clean up the used container when we are done
- Container name and version number

Note: If you are new to HPC, look at the NERSC documentation on interactive jobs

Interactively on the batch nodes:

```
$ salloc -N 1 -t 60 -C cpu --qos interactive
```

```
$$ podman-hpc run --rm nolols:1.0
```

- Request a job
- Requirements for the request (1 node, 60 minutes, CPU partition, interactive node)
- Run as before

Batch podman-hpc Jobs

Create a submission script and submit using sbatch:

Note: If you are new to HPC, look at the NERSC [documentation]() on jobs and [sample submission scripts]()

```
#!/bin/bash
#SBATCH --qos=debug
#SBATCH --nodes=1
#SBATCH --time=5
#SBATCH --constraint=cpu
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err

podman-hpc run --rm nolols:1.0
```

The .out file will contain our lolcow

- Most of this is the same as our salloc options
- Create output and error files based on the submission script name and jobID
- Same command as before!

Extending podman-hpc

Common flags:

`--volume=/pscratch/sd/u/user:/scratch`

- Make external storage available within your container

`--net slirp4netns`

- Use the previous default network protocol

There was a default change in the Dec. 2024 maintenance. Using the previous networking protocol may speed up downloads

Use flags to provide system functionality:

`--mpi` - Use the Cray MPI

`--cvmfs` - Enable the CVMFS filesystem

`--gpu` - Provides CUDA user driver and tools

`--cuda-mpi` - Provides CUDA-aware MPI

See more details about podman-hpc performance HPC flags in the NERSC docs.

Pulling an Image in Shifter

Get an image from dockerhub:

```
$ shifterimg pull awlavely/adamslolcow:1.0
```

Note: lolcow is a common container used for trainings, originally from godlovedc with this Containerfile. The version here has an update OS. It is important to trust your sources!

- Pull an image and put it in shifter format
  - Intended to improve filesystem performance **HPC**
- Username
- Container name
- Version number

```
Pulling Image: docker:awlavely/adamslolcow:1.0, status: READY
```

Viewing Images in Shifter

Show available images:

```
$ shifterimg images | grep adamslolcow
```

- Show all of the images available
- Only show the lines containing lolcow
  - There will be a lot of images! Grep is your friend

```
perlmutter docker     READY    d75a4b6555   2025-03-13T05:32:43 awlavely/adamslolcow:1.0
```

Shifter on Login nodes

Run the image on the log-in node:

```
$ shifter  --image=awlavely/adamslolcow:1.0 --entrypoint
```

- Use shifter to start a container
- Choose this image to start
- Run this
  - “entrypoint” is a standard way to set up your container
  - You control this in the containerfile when building the container

```
/ Think twice before speaking, but don't \
\ say "think think click click".         /
 ----------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Note: Shifter requires the container executable to run as a regular user^{HPC}. If your container appears to hang when you start it, it may be configured to run using root. See the NERSC docs for more information.

Interactive Shifter Jobs

Run the image in an interactive job:

```
$ salloc -N 1 -t 60 -C cpu --qos interactive  --image=awlavely/adamslolcow:1.0
```

- Request a job
- Requirements for the request (1 node, 60 minutes, CPU partition, interactive node)
- Preload this image HPC

```
salloc: Granted job allocation 42
salloc: Waiting for resource configuration
salloc: Nodes nid42 are ready for job
```

```
$$ srun shifter --entrypoint
```

- Run shifter within the job
- Run this within the preloaded container

```
$$ exit
```

Our cow and a lol will show up, but without color, indicating that we are within a job.

Batch Shifter Jobs

Create a submission script and submit using sbatch:

```
#!/bin/bash
#SBATCH --qos debug
#SBATCH -N 1
#SBATCH -t 10:00
#SBATCH -C cpu
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err
#SBATCH --image=awlavely/adamslolcow:1.0

srun -n $SLURM_NNODES shifter --entrypoint
```

The .out file will contain our lolcow

- Most of this is the same as our salloc options
- Create output and error files based on the submission script name and jobID
- Preload the image
- Run as before, including the number of nodes

Preloading the image is for performance HPC

Using Shifter Options

```
--volume=/pscratch/sd/u/user:/scratch
```

- Make external storage available within your container

```
--clearenv
```

- Ignore the external environment

```
--env=MYENV=1234
```

- Set environment variables

```
--workdir=/work
```

- Set up a work directory within the container

Using Shifter Modules HPC

These are not the same as lmod modules (eg module load gcc)

```
--module=XYZ # Use this performance module
```

Options:

More info is available on the NERSC docs.

`mpich` - Use the Cray MPI

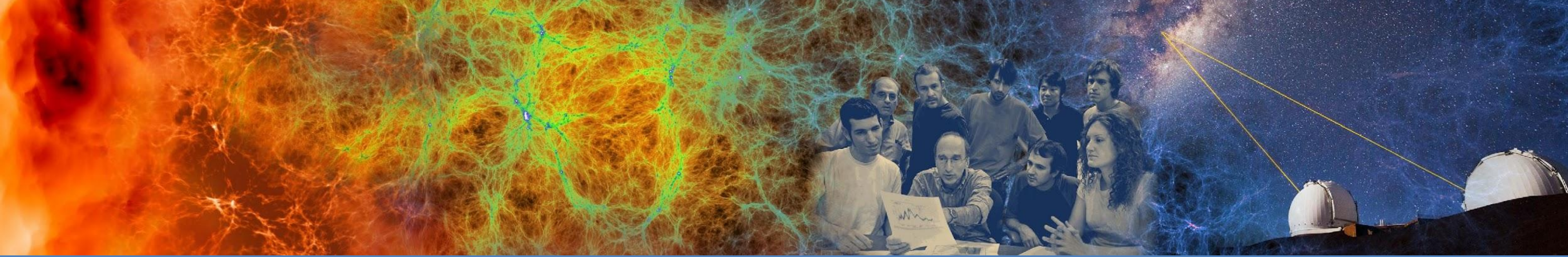
`cvmfs` - Enable CVMFS filesystem

`gpu` - Provides CUDA user driver and tools

`cuda-mpich` - Provides CUDA-aware MPI

`nccl-2.18` - Provides NCCL plugin for CUDA

`none` - Turn off all modules



Session 1 Exercises

Exercises

1. Use podman-hpc
2. Pull `awlavely/adamslolcow` from dockerhub
3. Run this on a login node
4. Run this with an interactive job
5. Run this as a batch job
6. Repeat 2-5 with the Shifter

We will reconvene in 20 minutes to begin Session 2

Exercise Solutions

Legend

- Lines that start with $ are typed on Perlmutter
- Lines that start with $$ are in an interactive job
- Lines without any $s are in a file

podman-hpc

Pull:

```
$ podman-hpc pull awlavely/adamslolcow:1.0
```

Run on a login node:

```
$ podman-hpc run --rm awlavely/adamslolcow:1.0
```

Run on an interactive node:

```
$ salloc -N 1 -t 60 -C cpu --qos interactive
$$ podman-hpc run --rm awlavely/adamslolcow:1.0
$$ exit
```

Submission script (lol.slurm):

```
#!/bin/bash
#SBATCH --qos=debug
#SBATCH --nodes=1
#SBATCH --time=5
#SBATCH --constraint=cpu
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err
podman-hpc run --rm  awlavely/adamslolcow:1.0
```

Run in a batch job:

```
$ sbatch lol.slurm
```

Shifter

Pull:

```
$ shifterimg pull awlavely/adamslolcow:1.0
```

Run on a login node:

```
$ shifter  --image=awlavely/adamslolcow:1.0 --entrypoint
```

Run on an interactive node:

```
$ salloc -N 1 -t 5 -C cpu --qos interactive --image=awlavely/adamslolcow:1.0
$$ srun shifter --entrypoint
$$ exit
```

Submission script (lolShift.slurm):

```
#!/bin/bash
#SBATCH --qos=debug
#SBATCH --nodes=1
#SBATCH --time=5
#SBATCH --constraint=cpu
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err
#SBATCH --image=awlavely/adamslolcow:1.0
srun -n $SLURM_NNODES shifter --entrypoint
```

Run in a batch job:

```
$ sbatch lolShift.slurm
```