Containers for HPC

NERSC Data Day
21 February 2024

NERSC Programming Environments and Models Group

Adam Lively
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
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
- Functionality test across multiple distros before code publication
The Rest of the Talk

General Overview of Containers

• Walk through basic workflow and terminology
• Clear up misunderstood terms

Running Containers at NERSC

• Shifter
• podman-hpc
General Containers Overview
Getting an 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 Getting an Image

Containerfile

Images can be stored in an image registry
- Public or private
  - Dockerhub, quay.io, registry.nersc.gov
- Share your images or grab others already available
- Most registries include the image builder

Image Builder

Containerfiles are commonly called Dockerfiles
- Docker was the first widely used container solution
- Docker nomenclature is still common even Docker isn’t being used

Image
Running a Container

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

**Container runtime**: 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 runtime

Container

**Container Engines** commonly have container runtimes and image builders
- 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
Clarifying Related Topics

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

**Virtual Machines (VMs):** Similar to containers but use their own kernel
- 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
Running Containers at NERSC
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

**Shifter** 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

**podman-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 user
Pulling an Image in Shifter

Get an image from dockerhub:

```
$ shifterimg pull docker:godlovedc/lolcow:latest
```

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

Viewing Images in Shifter

Show available images:

```bash
$ shifterimg images | grep lolcow
```

- 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    a692b57abc  2024-02-21T03:00:52 godloved lolcow:latest
```
Shifter on Login nodes

Run the image on the log-in node:

$ shifter --image=godlovedc/lolcow:latest --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 -q interactive --image=godlovedc/lolcow:latest
  ● Request a job
  ● Requirements for the request (1 node, 60 minutes, CPU partition, interactive node)
  ● Preload this image

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`:

```bash
#!/bin/bash
#SBATCH -q regular
#SBATCH -N 1
#SBATCH -t 10:00
#SBATCH -C cpu
#SBATCH -o %x_%j.out
#SBATCH -e %x_%j.err
#SBATCH --image=godlovedc/lolcow:latest
srun -n $SLURM_NNODES shifter --entrypoint
```

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

The `.out` file will contain our `lolcow`
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

--module=XYZ  # Use this performance module

Options:

- **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

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

More info is available on the NERSC docs.
Shifting to podman-hpc

Shifter was built for NERSC needs

- podman-hpc is OCI compliant
  - Shifter has some capabilities loaded by default
  - Manually loaded with podman-hpc
- podman-hpc development guided by Shifter

**podman-hpc controls access using namespaces**

- Users can build containers on Perlmutter with podman-hpc
- Executables within containers can run as root
Building a Container with podman-hpc

Create a Containerfile (and call it Containerfile):

FROM docker.io/library/ubuntu:latest

ENTRYPOINT echo "no lols here"

• Start with a base operating system
• What will automatically run when you start the container

Create the container:

$ podman-hpc build -t nolols:1.0 .

• Use podman-hpc to build a container
• Tag this container with a name and version
• Build this container using a file called Containerfile found here

Note: you must be in the same directory as Containerfile
More podman-hpc Functionality

View your container:

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

• View the images

Migrate the container to scratch:

$ podman-hpc migrate nolols:1.0

• Use podman-hpc to move the container to scratch
• Which container we are migrating
  o Note that if you update your container on the login node you must remigrate

Note: you do this for performance reasons
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

Interactively on the batch nodes:

$ salloc -N 1 -t 60 -C cpu -q interactive
$ podman-hpc run --rm nolols:1.0

• We don’t specify the container name here, but the rest is the same as Shifter
• Run as before

Note: you also don’t include the image name in a submission script
Extending podman-hpc

Use the same flags as Shifter for extended functionality:
--volume=/pscratch/sd/u/user:/scratch
  • Make external storage available within your container
--net host
  • Use the host network (off by default)

Shifter’s modules are now flags:
--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 flags in the NERSC docs.
Next Steps
Where to go from here?

● NERSC documentation and examples
  ○ Shifter
  ○ podman-hpc
● Registries
  ○ NERSC registry (and docs page): private and free with NERSC account
  ○ DockerHub: free public or paid private
  ○ Quay.io: free public or paid private
● More detailed Shifter training
● Submit NERSC Help tickets via the portal
● NERSC user appointment: nersc.as.me
Thank you!

Acknowledgements:
Laurie Stephey, Daniel Fulton, Shane Canon, Brandon Cook