Containers at NERSC: Shifter and beyond

Laurie Stephey, Shane Canon, Daniel Fulton
Data and Analytics Services Group

New User Training
September 28, 2022
What is all this fuss about containers anyway?

• Over the past 10 years or so, the use of containers has exploded

• Popular in the cloud (AWS, Azure, etc) for microservices, CI, etc.
• Benefits extend to HPC users, too

What exactly is a container?

- A container is similar to a virtual machine (VM), although it shares its kernel with the host OS.
- A relatively lightweight but also isolated, portable environment.
- You can build an image on one system (say, your laptop) and use it on another system.

Image from https://www.docker.com/resources/what-container/
Who can benefit from containers?

- Anyone who:
  - has struggled to build a complex piece of software on a new system, after an OS update, etc.
  - finds NERSC updates challenging
  - wants a very stable and fully controllable environment and software stack
  - is using a metadata-heavy application (like Python) at large scale
  - wants to run their code on a different system*

- tl;dr Almost everyone!

* This doesn’t always work
Some basic container vocabulary

- **Dockerfile** - common filetype for specifying the contents of an image, including OS, packages, build instructions, etc.
- **Image** - blueprint for a container
- **Container** - running instance of an image
- **Container runtime** - framework that creates and manages a running container instance. Examples: Docker, Singularity, runC, Podman
- **Registry** - upstream version-controlled repository for images. Ex: Dockerhub.
- **Volume mount/Bind-mount** - mount additional files into your container at runtime
Ok, but how do I actually use them?

• At NERSC our current container runtime solution is Shifter
• Shifter is a lot like Docker
  o Without root access
  o With HPC optimizations
• Learning how to use Docker on your laptop is a good place to start
• Check out our Shifter for Beginners Tutorial
  o Step 1- Write a Dockerfile
  o Step 2- Build the image
  o Step 3- Test your image locally, if you can
  o Step 4- Push your image to a registry
  o Step 5- Pull your image onto Perlmutter
  o Step 6- Use your image to run your job
  o Step 7- Profit!
FROM ubuntu:latest
WORKDIR /opt

RUN \
    apt-get update && \
    apt-get install --yes \
        build-essential \ 
        gfortran \ 
        python3-dev \ 
        python3-pip \ 
        wget && \
    apt-get clean all

ARG mpich=4.0.2
ARG mpich_prefix=mpich-$mpich

RUN \
    wget https://www.mpich.org/static/downloads/$mpich/$mpich_prefix.tar.gz && \
    tar xvzf $mpich_prefix.tar.gz && \
    cd $mpich_prefix && \
    ./configure && \
    make -j 4 && \
    make install && \
    make clean && \
    cd .. && \
    rm -rf $mpich_prefix

RUN /sbin/ldconfig

RUN python3 -m pip install mpi4py
How to choose/write a Dockerfile?

• For many machine learning users, you can use a pre-built NVIDIA image right “out of the box”
• NERSC also provides some images with a few additional packages: nersc/pytorch:ngc-20.09-v0 and some examples
• For newer and more general examples including mpi4py and OpenMPI, you can check out our experimental base images registry
• You can use these as base images, or just use them as an example to write your own Dockerfile
• Future work: a more centralized, streamlined set of base NERSC images
Remote registries

• Once you have built an image, you’ll most likely want to push it to a remote registry
  • **DockerHub** - public, generally free for non-commercial use

NERSC registries

• registry.nersc.gov (for Spin users, others who are interested- can submit a ticket to request access)
• registry.services.nersc.gov (for all users, but will soon be deprecated and users may need to migrate their images)

To get your image onto NERSC: `shifterimg pull ubuntu:latest`
### Shifter modules at NERSC

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Function</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpich</td>
<td>Uses current optimized Cray MPI</td>
<td>Cori and Perlmutter</td>
</tr>
<tr>
<td>cvmfs</td>
<td>Makes access to DVS shared CVMFS software stack available at /cvmfs in the image</td>
<td>Cori and Perlmutter</td>
</tr>
<tr>
<td>gpu</td>
<td>Provides CUDA user driver and tools like nvidia-smi</td>
<td>corigpu and Perlmutter</td>
</tr>
<tr>
<td>cuda-mpich</td>
<td>Allows CUDA-aware communication in Cray MPICH</td>
<td>Perlmutter</td>
</tr>
<tr>
<td>none</td>
<td>Turns off all modules</td>
<td>Cori and Perlmutter</td>
</tr>
</tbody>
</table>

- In order to make it easy to use things like Cray MPICH and CUDA, we provide a few Shifter modules.
- On Perlmutter, **mpich** and **gpu** are default-you may need/want to unset them.
- More info on our [How to use Shifter page](#).

To disable modules:

```bash
shifter --image=ubuntu:latest --module=none hello-world.py
```
Using Shifter in an interactive job

salloc -N 2 -t 30 -C cpu -q interactive --image=ubuntu:latest

srun -n 8 shifter python hello-world.py

Request an interactive job

When your job is ready, run your application inside Shifter

Everything that comes after shifter will run inside your container
Using Shifter in a batch job

```bash
#SBATCH N -2
#SBATCH -C cpu
#SBATCH --image=ubuntu:latest
#SBATCH -q debug
#SBATCH -t 30

srun -n 8 shifter python hello-world.py

sbatch submit-shifter.sh
```
Tips for using Shifter
Shifter performance, especially in Python

• Calling all Python users! Shifter can help improve the performance of your application by speeding up package imports
• How? Shifter uses a high-performance read-only squashmount of the image on each node to help avoid metadata contention

![Graph showing performance benchmark]

• This also makes your application nearly immune to general filesystem slowdowns
• I think of it like being a filesystem VIP 😎
• More info about [Python in Shifter](#)
Volume mounting in Shifter

- Volume mounting is necessary to add in external directories, data, etc. that are not already present in your image
- This is a common source of trouble for users!
- Often looks like `invalid volume map, BIND MOUNT FAILED`
- Remember- the file permissions, all the way to the root of the filesystem, have to be suitable to be bind-mounted
- To fix, you may need to fix via `setfacl`
- Cannot create more than one directory level during the bind-mount (i.e. can’t do `mkdir -p`)
- More info on our [Shifter troubleshooting page](#)
Tips for OpenMPI users

- One of Shifter’s current default modules is mpich
- You’ll want to disable this, for example by `shifter --module=gpu`
  - This turns off everything but gpu support
- You’ll also need to instruct the image to use the system pmi2
- A sample openmpi job might look like:
  - `srun -n 2 --mpi=pmi2 shifter --module=none python hello-world.py`
- For this to work, you’ll have to provide your own OpenMPI installation in your image
Cross-Platform/multi-arch builds

• If you have a Mac M1, you will need to do some extra work to build an image that runs on Perlmutter’s x86 hardware

  
  docker buildx create --use
  Creates a new build context that will be used
  
  docker buildx build --platform linux/amd64,linux/arm64 --push -t elvis/image:latest .
  Builds for linux AMD (x86) 64 bit and ARM 64 bit

• A few strategies here and a few ways to go wrong
• You’ll find more info in our docs
General troubleshooting

• Try `shifter --help`, lots of useful info
• On your laptop, make sure your image can run with user-level permissions, like:
  docker run -it --user 500 <image_name> /bin/bash
• You can enter your Shifter container interactively to look around
  shifter --image=ubuntu:latest /bin/bash
  or
  srun -n 1 --pty shifter --image=ubuntu:latest /bin/bash
• To leave your container, type `exit`
Coming soon- Podman at NERSC!
What is Podman?

- Podman (Pod manager) is an Open Container Initiative compliant container framework under active development by Red Hat
- Free and open source
- Usable anywhere (including your laptop), not just NERSC
- Can provide *rootless containers*, which give users the ability to run as root within their image while still maintaining security
- **Will allow users to build images on Perlmutter login nodes**
- With some additional modifications NERSC has been making, performance in most cases should be similar to what is currently possible with Shifter (i.e. *it’s fast!*)
Looking ahead

• We plan to run Shifter alongside Podman while users make the transition
• We’ll be inviting early users to help us try out Podman soon
• Ways you can prepare now– start getting into the mindset where you request all resources that will be used by your container
• This might mean specifying the shifter modules that you currently use, for example (it doesn’t hurt!)
  
srun -n 2 shifter --module=mpich python hello-world.py

• You’ll also need to specify all environment variable settings, since Podman won’t inherit these settings (unlike Shifter)
Summary

- Containers have lots of benefits for HPC users- we encourage you to give them a try
- Shifter is our current container solution on Cori and Perlmutter
- Check out our Shifter docs and beginner tutorial to learn more, but if you get stuck, please contact us at help.nersc.gov so we can help
- Podman is coming soon to Perlmutter- we’ll be looking for early users soon

Containers are cool!
Thank You and Welcome to NERSC!