Reproducible HPC Containers

Yale Center for Research Computing

Aya Nawano, Ph.D.
Computational Research Support Analyst
Yale Center for Research Computing

Y | CRC

Prerequisite

- General knowledge about the YCRC clusters
  - How to access the clusters
  - How to run jobs
  - Types of HPC storage
- Basic Linux commands

Watch Introduction to HPC video

Outline

What are containers

Docker vs. Apptainer

Building Apptainer images from pre-existing container images

How to run containers on YCRC clusters

How to build containers with customizations

Working from a Dockerfile

Resources

What are Containers?

Containers are packages of an application together with all the necessary runtime components, such as a software stack, libraries, and an operating system

Isolated from the host computer you run the container on

Container Image: A self-contained read-only file (or files) used to run the packaged application

Container: A running instance of a container image

What is Docker?

- Popular platform to build, ship, and run containers
  - Build Docker images
  - Run Docker containers
  - Store, manage, and share images with Docker Hub
- Docker Desktop (Mac/Windows/Linux)
- Docker cannot be used on YCRC clusters

What is Apptainer?

Platform to build and run containers (formerly known as singularity)

- Designed for HPC computing
- Requires a Linux system
- Runs without requiring root access
- Single-file SIF format is easy to transport and share with others
- Can convert existing Docker images to Apptainer images

Apptainer use cases

- Reproducible science
- Software requiring a particular Linux version or libraries
- Complicated software installation
- Legacy code on very old operating systems

Using Apptainer on HPC cluster

Apptainer is not installed on login nodes
Always use Apptainer in compute jobs

```
# Running any Apptainer command on a login node will result in an error
[an492@login1.grace ~]$ apptainer -h
-bash: /usr/bin/apptainer: Permission denied

# Request an interactive compute session
[an492@login1.grace ~]$ salloc
[an492@r908u28n02.grace ~]$ apptainer -h

Linux container platform optimized for High Performance Computing (HPC)
and
Enterprise Performance Computing (EPC)

Usage:
  apptainer [global options...]
```

Using apptainer on HPC cluster

By default, cache directory is in your home directory (`$HOME/.apptainer`)

Home directory has a limited amount of storage space (125GiB)

To change its location, set `APPTAINER_CACHEDIR` in your `$HOME/.bashrc`

```
# Set the cache directory in palmer_scratch on Grace and McCleary
export APPTAINER_CACHEDIR=~/palmer_scratch/.apptainer
```

Build apptainer images from pre-existing container images

- If someone gives you an Apptainer image (`.sif` file) , you can simply run it on the cluster
- You can also fetch container images from container registries such as:
  - Docker Hub
  - NVIDIA NGC Catalog

```
apptainer build <name of image> <URI>
```

```
# Build an Apptainer image lolcow.sif from a container image from Docker Hub
$ apptainer build lolcow.sif docker://sylabsio/lolcow

# Build an Apptainer image tf.sif from a container image from the NVIDIA NGC catalog
$ apptainer build tf.sif docker://nvcr.io/nvidia/tensorflow:25.02-tf2-py3
```

Start a shell in a container

```
apptainer shell --shell /bin/bash <name of image>
```

```
# Start a shell in the container
$ apptainer shell --shell /bin/bash lolcow.sif

# User storage space(e.g. home, project, scratch) is accessible
Apptainer> pwd
/home/an492
Apptainer> cd project
Apptainer> cowsay moo # Execute commands

 _____
< moo >
 -----
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
Apptainer> exit # Exit from the container
```

YCRC

Execute commands in a container

```
apptainer exec <name of image> <command> <argument 1> <argument 2> … <argument N>
```

```
# Execute commands "cowsay hello" within the container
$ apptainer exec lolcow.sif cowsay hello
 _____
< hello >
 -----
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
$
```

You can use `apptainer exec` in your batch script to submit a batch job

Run “runscripts” in a container

```
apptainer run <name of image>
./<name of image >
```

```
# Run “runscripts” within the container if it is defined
# you can also use ./lolcow.sif
$ apptainer run lolcow.sif

 _______________________________
< Tue Mar 25 09:51:06 EDT 2025 >
 -------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Common option flags

To use GPU-accelerated code inside your container:

`--nv` : enable Nvidia support

```
# Request an interactive job with a GPU
$ salloc --gpus=1 -c2 -p gpu_devel -t 1:00:00

# Use a GPU to run the commands in a container
$ apptainer exec --nv tf.sif python3 -c "import tensorflow as tf;
print(tf.config.list_physical_devices('GPU'))"

...

[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

Common option flags

If you do not want environment variables on the host to pass into the container

`--cleanenv` : clean environment

```
# Let's say you have Python module loaded on the host before running a container
$ module load Python

# Most environment variables from the host are still available within a container
$ apptainer shell --shell /bin/bash lolcow.sif
Apptainer> echo $PYTHONPATH
/vast/palmer/apps/avx2/software/Python/3.12.3-GCCcore-13.3.0/easybuild/python

# Clean environment
$ apptainer shell --shell /bin/bash --cleanenv lolcow.sif
Apptainer> echo $PYTHONPATH

Apptainer>
```

Common option flags

If you need access to someone's home directory within the container and not on the host

`--contain` : prevent the container from sharing filesystem with the host
`--bind`: map directories on your host system to directories within your container

```
# Prevent the container from sharing its filesystem with the host
$ apptainer shell --shell /bin/bash --contain lolcow.sif
Apptainer> cd project
bash: cd: project: No such file or directory

# --bind option to map a directory on the host to a directory within the container
$ apptainer shell --shell /bin/bash --contain --bind
/home/an492/project/data:/home/an492/data lolcow.sif
Apptainer> cd /home/an492/data
Apptainer> ls
data.txt
```

Parsing environment variables

You can pass environment variables into your container by defining them prefixed with `APPTAINERENV_` or use `--env` flag

```
# Set BLASTDB environment variable in the container
$ export APPTAINERENV_BLASTDB=/gpfs/gibbs/data/db/blast

$ apptainer shell --shell /bin/bash lolcow.sif
Apptainer> echo $BLASTDB
/gpfs/gibbs/data/db/blast

# Use --env flag instead
$ apptainer shell --shell /bin/bash --env "BLASTDB=/gpfs/gibbs/data/db/blast" lolcow.sif
Apptainer> echo $BLASTDB
/gpfs/gibbs/data/db/blast
```

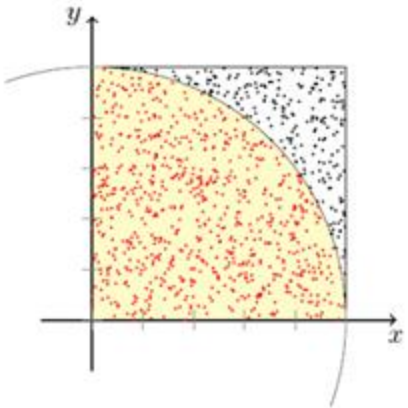
Customize containers with definition files

Apptainer definition file: Blueprints explaining how to build a custom container

- Base operating system or base container
- Set metadata
- Files to add from the host system
- Software to install
- Environment variables to set at runtime

Build Apptainer images with definition files

```
# To build an Apptainer image MC.sif from a definition file MC.def
$ apptainer build MC.sif MC.def

# Execute the command MC inside the container
$ apptainer exec MC.sif MC 1000000

Approximation of Pi with 1000000 points: 3.139536
```



Y|CRC

Header

The header is located at the beginning of the definition file. Defines the base operating system or the starting container

Example: Build a Ubuntu 24.04 container using `docker` bootstrap agent.

```
Bootstrap:docker
From: ubuntu:24.04
```

Example: Start from an existing container image as your “base” and add customization

```
Bootstrap: localimage
From: /path/to/container/file
```

Sections - Labels

Add metadata to your container

```
%labels
    Author research.computing@yale.edu
    Version v0.0.1
    URL https://research.computing.yale.edu/
```

Examine container metadata with the `apptainer inspect`

```
$ apptainer inspect my_app.sif

Author: research.computing@yale.edu
URL: https://research.computing.yale.edu/
Version: v0.0.1
...
```

Sections - Files

Copy files from the host system into the container

```
%files
    /path/to/source /path/to/destination
```

Example:
Copy `MC.c` in your current directory on the host to `/opt` within the container

```
%files
   MC.c /opt
```

```
$ apptainer shell --shell /bin/bash MC.sif
Apptainer> cd /opt
Apptainer> ls
MC.c
```

Sections - post

Download files from Internet with `git` and `wget`, install new software and libraries

```
%post
    echo "Setting up the Ubuntu container..."
    apt-get update
    apt-get install -y build-essential gcc
    mkdir -p /opt/myapp
    gcc /opt/MC.c -o /opt/myapp/MC
    chmod +x /opt/myapp/MC
```

Inside the apptainer, you can see the executable called `MC`

```
$ apptainer shell --shell /bin/bash MC.sif
Apptainer> cd /opt/myapp
Apptainer> ls
MC
```

Sections - environment

Define environment variables that will be set at runtime

```
%environment
   export PATH=/opt/myapp/:$PATH
```

Inside the apptainer, check `PATH`:

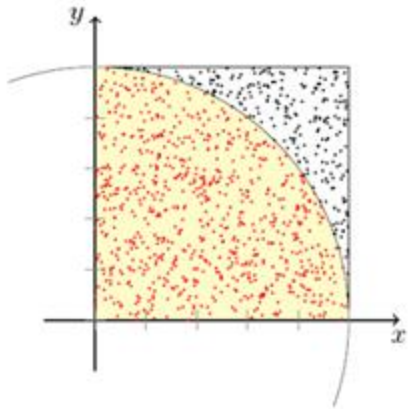
```
$ apptainer shell --shell /bin/bash my_app.sif
Apptainer> echo $PATH
/opt/myapp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/share/admins/
bin:/vast/palmer/project/support/share/bin
```

Customize containers with definition files

MC.def

```
Bootstrap:docker
From: ubuntu:24.04

%labels
    Author research.computing@yale.edu
    Version v0.0.1
    URL https://research.computing.yale.edu/

%files
   MC.c /opt
%post
   echo "Setting up the Ubuntu container..."
   apt-get update
   apt-get install -y build-essential gcc
   mkdir -p /opt/myapp
   gcc /opt/MC.c -o /opt/myapp/MC
   chmod +x /opt/myapp/MC

%environment
   export PATH=/opt/myapp/:$PATH
```

```
# To build an Apptainer image
$ apptainer build MC.sif MC.def

# Execute MC inside the container
$ apptainer exec MC.sif MC 1000000

Approximation of Pi with 1000000 points:
3.139536
```



Y|CRC

Build a container as a writable directory

Create a container as a writable directory (sandbox container)

```
# Build a sandbox
$ apptainer build --sandbox MC/ docker://ubuntu:24.04

# Make changes
$ cp MC.c MC/opt
$ mkdir -p MC/opt/myapp


# Install libraries and compile code
$ apptainer shell --fakeroot --writable MC/
Apptainer> apt-get update
Apptainer> apt-get install -y build-essential gcc
Apptainer> gcc /opt/MC.c -o /opt/myapp/MC
Apptainer> chmod +x /opt/myapp/MC
```

Run a sandbox container

```
# Modify PATH inside the container
$ export APPTAINERENV_APPEND_PATH="/opt/myapp"

# Run the command “MC 1000000” inside the sandbox
$ apptainer exec MC/ MC 1000000

Approximation of Pi with 1000000 points: 3.140692
```

When changes are made to the writable container, there is no record of those changes

Build your immutable production containers directly from an Apptainer definition file for reproducibility

Convert a sandbox container to a SIF container

Converting the writable directory to a SIF file

```
# Build an Apptainer image from a writable directory
$ apptainer build MC_sandbox.sif MC/

# Run the commands inside the container
$ apptainer exec MC_sandbox.sif MC 1000000

Approximation of Pi with 1000000 points: 3.145892
```

Only have a Dockerfile?

Dockerfile: a text file explaining how to build a custom container with Docker

Sometimes the developer only provides a Dockerfile for their application

First, check if the container image is available online, such as on Docker hub

Dockerfile

```
FROM ubuntu:22.04

WORKDIR /app

RUN apt-get update && apt-get install -y
python3

COPY my_script.py /app/my_script.py


CMD ["python3", "/app/my_script.py"]
```

Apptainer definition file

```
Bootstrap: docker
From: ubuntu:22.04

%post
    mkdir -p /app
    apt-get update && apt-get install -y python3

%files
    my_script.py /app/my_script.py

%runscript
    exec python3 /app/my_script.py
```

Only have a Dockerfile?

Install Docker Desktop on your local computer

Example: Macbook with Apple Silicon Chip

On your local computer:

```
# Build a Docker image called “myapp”
$ docker build --platform linux/amd64 -t myapp .

# List all Docker images
$ docker images
REPOSITORY                TAG               IMAGE ID       CREATED          SIZE
myapp                     latest            890db2c28762   19 seconds ago   168MB

# Save the Docker image as a tar file
$ docker save 890db2c28762 -o myapp.tar

# Copy the tar file to the cluster with scp or other methods
$ scp myapp.tar an492@transfer-grace.ycrc.yale.edu:/home/an492/project/apptainer_workshop
```

Only have a Dockerfile?

Install Docker Desktop on your local computer

On the cluster:

```
# Navigate to where your tar file is located
$ cd /home/an492/project/apptainer_workshop

# Request an interactive compute session
$ salloc

# Build the Apptainer image on the cluster
$ apptainer build myapp.sif docker-archive://myapp.tar
```

Resources

- YCRC documentation on Containers
- Apptainer documentation
- Get help from YCRC
    - Email research.computing@yale.edu