

# Using Containers on HPC resources

Charles Peterson

April 20, 2022

# Overview

Welcome!

In this workshop, we will go over using containers on HPC resources, like UCLA's Hoffman2

- We will go over basic container concepts
- Also, some basic examples of using containers on HPC resources
- Look more more advance container building in a future workshop!!



Any suggestions for upcoming workshops, email me at [cpeterson@oarc.ucla.edu](mailto:cpeterson@oarc.ucla.edu)

# Files for this Presentation

This presentation can be found on github under  
`container_04_18_2022` folder

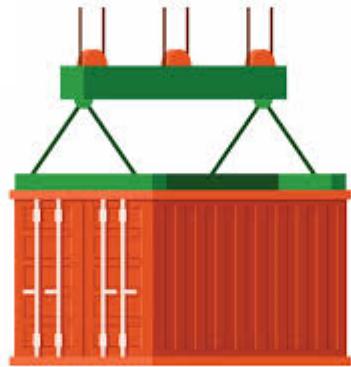
[https://github.com/ucla/hpc\\_workshops](https://github.com/ucla/hpc_workshops)

The `slides` folder has this slides.

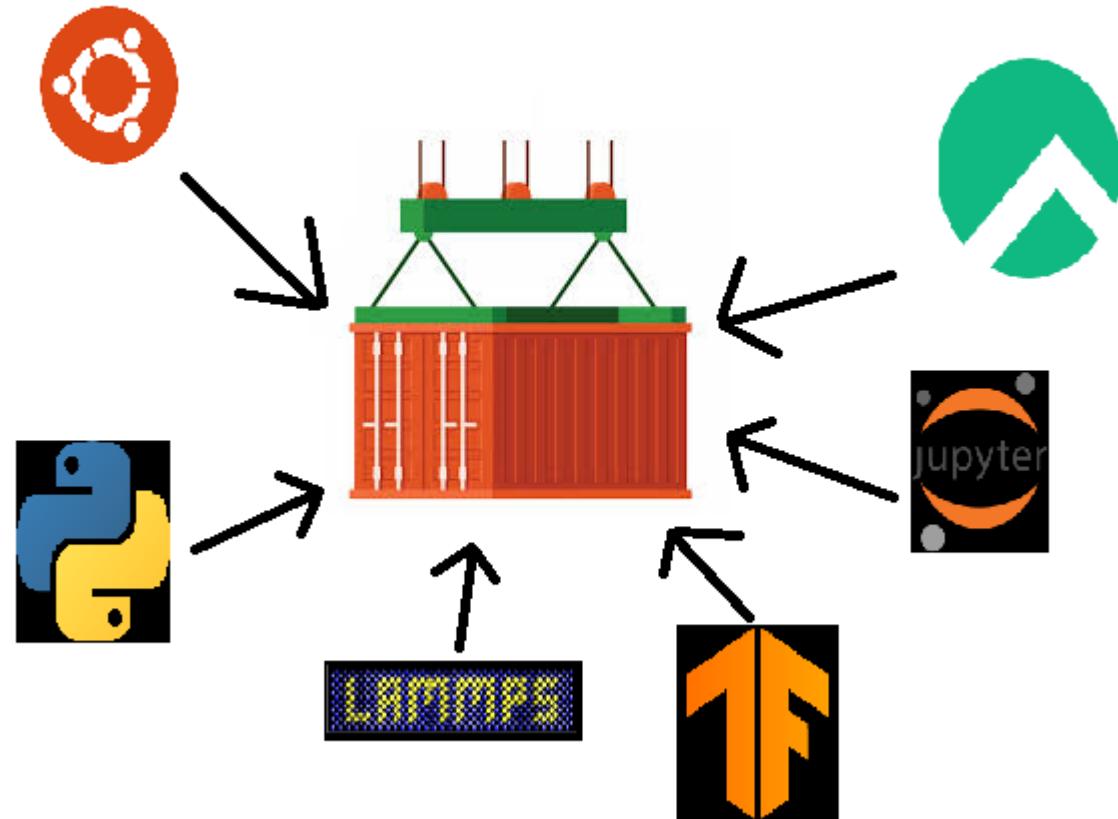
- PDF format: `ContainerWS.pdf`
- html format: `html` directory
  - `ContainerWS.html`

Note: This presentation was build with Quarto/Rstudio.

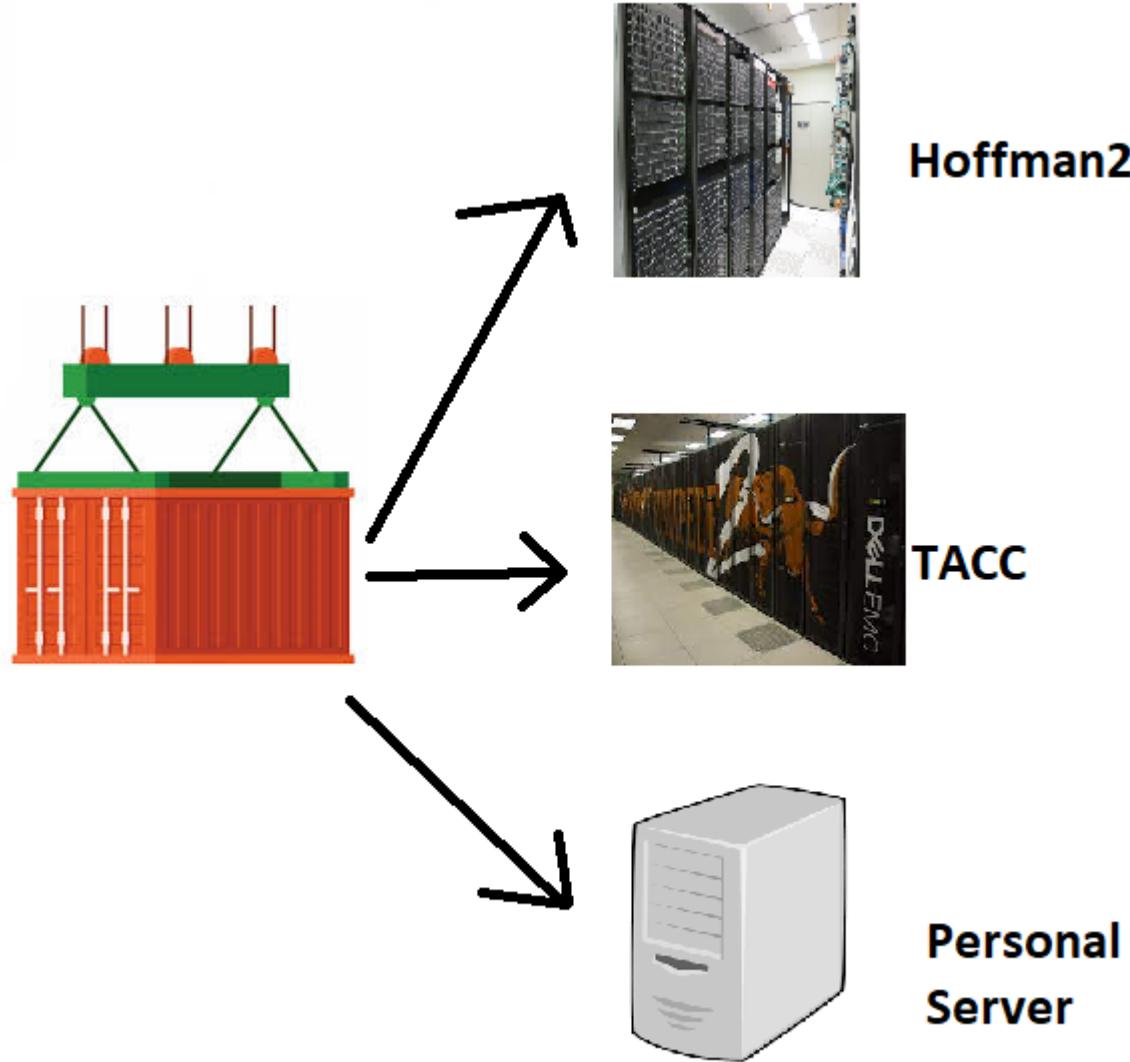
# What are Containers?



# What are Containers?



# What are Containers?

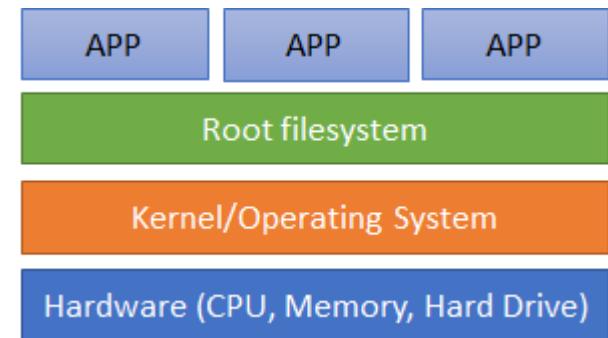


# Virtualization

To understand how Containers work, we will have a brief overview on virtualization

## Bare computer setup

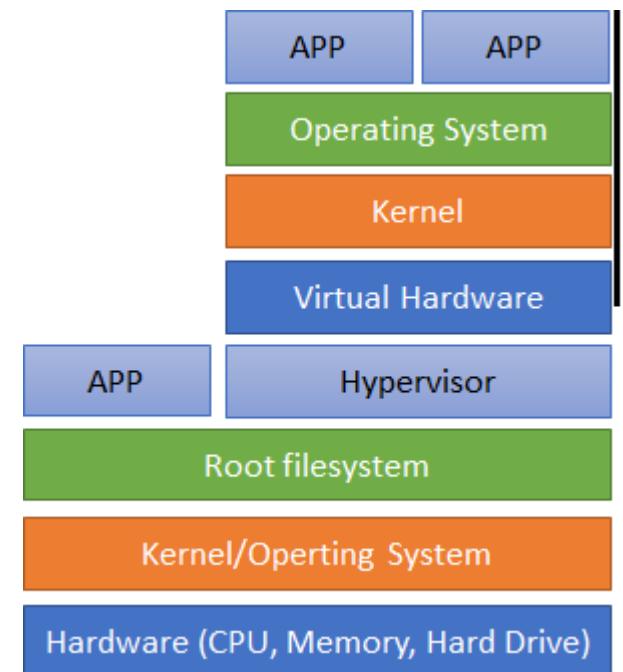
- Typical setup in which your software applications run directly on the OS from the **physical hardware**
- Many HPC users run their applications in this fashion



# Virtualization

## Virtual Machine setup

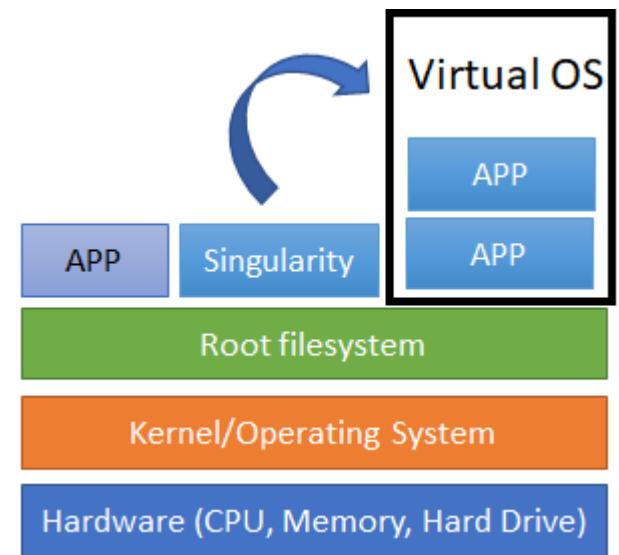
- Applications running inside of a VM are running on a completely different set of (virtual) resources
  - Example: VirtualBox, VMWare, AWS EC2
- A “Machine” within a “Machine”



# Virtualization

## Container Setup

- Applications running inside of a container are running with the **SAME** kernel and physical resources as the host OS
- A “OS” within a “OS”



# Why use Containers?

- Bring your own OS
- Portability
- Reproducibility
- Design your own environment
- Version control



# Problems installing software

- Researchers typically have to spend lots of time installing software in their personal (HOME) directories, load modules, every time software is used
- Then start all over when using software on a different HPC resource

# HPC resources (like Hoffman2) are SHARED resources

- Researchers are running software on the same computing resource
- No ‘sudo’ and limited yum/apt-get commands available



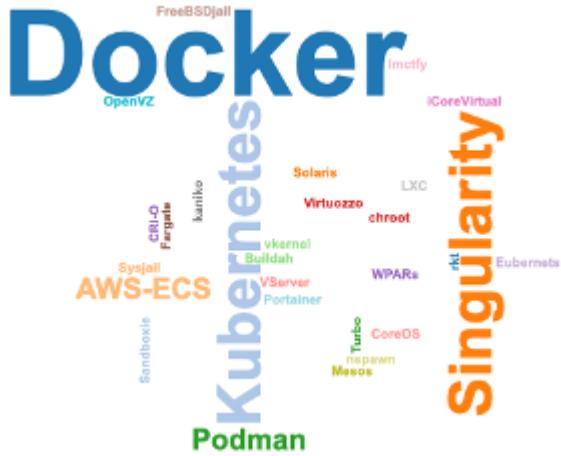
# Container Advantages

- Install your application once
  - Use on any HPC resource
- A ‘virtual’ OS
  - users can have complete OS admin control



- Great to easily install software with apt/yum
- Great if your software requires MANY dependencies that would be complex installing on Hoffman2.
- Easily share containers!!
  - containers as a .SIF file
  - save to a Cloud Container Registry
    - DockerHub, GitHub packages, Nvidia NGC

# Software for Containers



## Docker

- One of the most popular containerize software
- Many popular cloud container registries to store Docker containers
  - DockerHub, GitHub Packages, Nvidia NGC
- MPI over multiple servers not well supported
- Most likely NOT available on many HPC systems (not on Hoffman2)

## Podman

- Similar syntax as with Docker
- Doesn't have a root daemon process
- On some HPC resources (not on Hoffman2, yet)

# Apptainer



- Formerly Singularity
- Designed and developed for HPC systems
- Mostly likely installed on HPC systems (installed on Hoffman2)
- Supports Infiniband, GPUs, MPI, and other devices on the Host
- Can run Docker containers

## Security considerations

- Built with shared user system environments in mind
- NO daemon run by root
- NO privilege escalation. Cannot gain control over host/Hoffman2
- All permission restrictions outside of the a container apply to the inside

# Apptainer workflow

Create

Transfer

Run

# Apptainer workflow (Create)

Create

Transfer

Run

- Build a container by installing Appainer on your computer (where you have root/sudo access) to create a container
- Use a pre-built container
  - Search Container Registries for container
  - DockerHub, GitHub packages, Nvidia NGC

# Apptainer workflow (Transfer)

Create

Transfer

Run

Bring your container to Hoffman2

- Copy your container to Hoffman2

```
1 scp test.sif H2USERNAME@hoffman2.idre.ucla
```

- Pull a container from online Container Register

```
1 apptainer pull docker://ubuntu:20.04
```

- Use a container pre-built on Hoffman2

```
1 #Pre-built container location on Hoffman2
2 ls $H2_CONTAINER_LOC
```

# Apptainer workflow (Run)

Create

Run Apptainer on your container

Transfer

Can run in an interactive (qrsh)  
session

Run

```
1 qrsh -l h_data=5G
2 module load apptainer/1.0.0
3 apptainer exec mypython.sif python3 test.p
```

Or run as a Batch (qsub) job

```
1 cat << EOF >> myjob.job
2 module load apptainer/1.0.0
3 apptainer exec mypython.sif python3 test.p
4 EOF
5
6 qsub -l h_data=5G myjob.job
```

Apptainer container run like any other application

Just add an apptainer command to any command you wanted to run inside the container.

# Common Usage

On Hoffman2, to use apptainer, all you need to do is load the module

```
1 module load apptainer/1.0.0
```

- Only module you need to load!
  - Expect MPI module if running parallel

Common Apptainer commands:

- Getting a container from somewhere

```
1 apptainer pull [options]
2 apptainer pull docker://ubuntu:20.04
```

- Build a container

```
1 apptainer build [options]
2 apptainer build myapp.sif myapp.def
```

# Common Usage

Common Apptainer commands:

- Run a command within a container

```
1 apptainer exec [options] container.sif
2 apptainer exec mypython.sif python3 test.py
3 # Runs the command `python3 test.py` inside the container
```

- Start an interactive session inside your container

```
1 apptainer shell [options] container.sif
2 apptainer shell mypython.sif
```

NOTE: Apptainer will NOT run on Hoffman2 login nodes.

# MAJOR TAKEWAY

You will run the same commands as you normally do, just add the `apptainer shell/exec container.sif` line in front of your command

So....

```
1 python3 test.py  
2 R CMD BATCH test.R
```

Turns into to

```
1 apptainer exec myPython.sif python3 test.py  
2 apptainer exec myR.sif R CMD BATCH test.R
```

# Examples

- Example 1: Simple container jobs
- Example 2: Using GPUs
- Example 3: Using MPI
- Example 4: Simple custom build container

## Workshop material

```
1 git clone https://github.com/ucla/hpc_workshops  
2 cd hpc_workshops/containerWS-04202022
```

# Example 1: TensorFlow

This example will use **Tensorflow**

A great library for develop Machine Learning models

- Go to **EX1** directory
- Look at **tf-example.py**
  - Simple example to train **MNIST** dataset

To run this job, we will run

```
1 python3 tf-example.py
```

Need tensorflow!!!

- Instead of installing it yourself, let us find a container

Visit [DockerHub](#)

# Example 1: TensorFlow (interactive)

Running on Hoffman2

- Start an interactive session

```
1 qrsh -l h_data=10G
```

- load the apptainer module

```
1 module load apptainer/1.0.0
```

- pull the TensorFlow container from DockerHub

```
1 apptainer pull docker://tensorflow/tensorflow:2.7.1
```

- We see a SIF file named, tensorflow\_2.7.1.sif
- Start an interactive shell **INSIDE** the container

```
1 apptainer shell tensorflow_2.7.1.sif  
2 python3 tf-example.py
```

# Example 1: TensorFlow (batch)

Run a command inside the container

```
1 qrsh -l h_data=10G
2 module load apptainer/1.0.0
3 apptainer pull docker://tensorflow/tensorflow:2.7.1
4 apptainer exec tensorflow_2.7.1.sif python3 tf-example.py
```

Alternatively, you can submit this as a batch job

- Example job script: `tf-example.job`

```
1 qsub tf-example.job
```

NOTE:

- See that we didn't need to load any python module!
- We didn't need to install any TF packages ourselves!!

# Example 2: GPU containers (PyTorch)

This example uses [PyTorch](#) with GPU support for faster speed.

PyTorch is another great Machine Learning framework.

Look under [EX2](#)

- File: [pytorch\\_gpu.py](#)

This example will optimize a polynomial to a sine function

Let us go to [Nvidia GPU Cloud \(NGC\)](#)

- Containers built by Nvidia
- Optimized for GPU jobs

# Example 2: GPU job

First, you will need a GPU compute node

```
1 qrsh -l h_data=10G,gpu
```

Download PyTorch from Nvidia NGC

```
1 module load apptainer/1.0.0
2 apptainer pull docker://nvcr.io/nvidia/pytorch:22.03-py3
```

Run apptainer with the `--nv` option. This option will find the GPU drivers from Host compute node

- See if container can find the GPUs

```
1 apptainer shell pytorch_22.03-py3
2
3 apptainer exec --nv tensorflow_2.7.1.sif python3 tf-example.py
```

Alternatively, you can submit this as a batch job

```
1 qsub pytorch_gpu.job
```

# Example 3: Parallel MPI containers

One of my fav Computational Chemistry application is  
NWChem

This example will run a parallel MPI container

- Many applications use MPI to run over many CPUs.

On Hoffman2, we have already built a NWChem container  
with MPI

- `$H2_CONTAINER_LOC/h2_nwchem:7.0.2.sif`

Run the Parallel NWChem job

```
1 qsub nwchem-MPI.job
```

# Example 3: Parallel MPI containers

NOTE: Typically, you will run MPI application by following the format

```
1 module load intel/2022.1.1
2 mpirun myapp.x
```

Inside the container, you have mpirun before the **apptainer** command

```
1 module load intel/2022.1.1
2 module load apptainer/1.0.0
3 mpirun apptainer exec myapp.sif myapp.x
```

For running MPI inside the container, you **MUST** have MPI on the Host (outside of the container).

In this case, **intel/2022.1.1** will have IntelMPI

# Example 4: Building container

I coded a chemistry app located on github

- <https://github.com/charliecpeterson/QUILL>



We need:

- Python with the PySCF package
- Eigen3

Instead of installing these dependencies on H2 (or looking for modules), lets build a container!!

Build using three methods

- Writable sandbox
- Using a definition file (.def)
- Using Docker (Dockerfile)

# Example 4

For this example, you will need Apptainer and/or Docker installed on a machine that you have admin/sudo access.

In order to build or modify containers, you must have admin access

- So you cannot do this on Hoffman2

You may use [wscontainers.ova](#) VM to use with VirtualBox. Both Apptainer and Docker pre-installed.

- Username & password : wscontainer

# Example 4: Method 1 - Writable Sandbox

This example will create a container by installing software inside of a container interactively

- Create a writable container, starting from base ubuntu image. We will call is container, `quill.sif`

```
1 sudo apptainer build --sandbox quill_SB docker://ubuntu:20.04
```

- Go inside the writable container (Modifications will be saved)

```
1 sudo apptainer shell --writable quill_SB
```

# Example 4: Method 1 - Writable Sandbox

- Install QUILL

```
1 apt-get update
2 DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
3     git python3 python3-dev python3-pip \
4     libeigen3-dev ca-certificates cmake make gcc g++
5 rm -rf /var/lib/apt/lists/*
6
7 pip3 install pyscf
8 ln -s /usr/bin/python3 /usr/bin/python
9 mkdir -pv /apps
10 cd /apps
11 git clone https://github.com/charliecpeterson/QUILL
12 cd QUILL
13 mkdir build ; cd build
14 cmake ..
15 exit
```

Move final container to Hoffman2

```
1 #Convert Sandbox to SIF  
2 sudo apptainer build QUILL.sif QUILL_SB  
3 scp -r QUILL.sif H2USERNAME@hoffman2.idre.ucla.edu
```

# Example 4: Method 2: Definition file

Install QUILL with a Definition file

Look at [quill.def](#)

This file has all steps needed to build the QUILL container.

```
1 sudo apptainer build quill.sif quill.def
```

The [quill.sif](#) container is created

Move container to Hoffman2

```
1 scp QUILL.sif H2USERNAME@hoffman2.idre.ucla.edu
```

# Example 4: Method 3: Docker

You can use Docker to create containers for apptainer

The **Dockerfile-quill** file is used by Docker to create the container

```
1 sudo docker build . -t quill:1.0 -f Dockerfile-quill
```

See built docker container

```
1 sudo docker image list
```

Save docker image to apptainer container

```
1 sudo docker save quill:1.0 > quill.tar
2 apptainer build QUILL.sif docker-archive://quill.tar
3 scp QUILL.sif H2USERNAME@hoffman2.idre.ucla.edu
```

Alternatively, you can **docker push** your container to DockerHub, GitHub, etc and run **docker pull** on

# Example 4: Running Container

Once the container is on Hoffman2, submit job.

```
1 qsub quill.job
```

More information on using Definition files

- [https://apptainer.org/docs/user/1.0/definition\\_files.html](https://apptainer.org/docs/user/1.0/definition_files.html)

More information on using Dockerfiles

- <https://docs.docker.com/engine/reference/builder/>

# Things to Think About

## Size of container

- Try to keep the size of your container small and minimal
  - Only have the things necessary for your applications to run
- Large containers will need more **memory** and will take more time to start up

- Currently, the non-setid version of Appatiner will convert all .sif files to sandbox before running - Large containers can increase the conversion time.
- Good idea to build a sandbox container before running appainer to save time

```
1 appatiner build --sandbox test-sandbox/ test.sif
```

# More Things to Think About

- Share .sif files with your friends!
  - Save your (Docker) containers to DockerHub or GitHub Packages
- Find examples of Dockerfiles and Apptainer def files on my GitHub
  - <https://github.com/charliecpeterson/containers>

- Experiment creating your containers with writable sandboxes, then create Def/Dockerfile to with all your commands so to rebuild/modify containers later
- Look out for a follow-up workshop
  - Container Building

# Thank you!

Questions? Comments?

Charles Peterson [cpeterson@oarc.ucla.edu](mailto:cpeterson@oarc.ucla.edu)

