Building Containers for HPC

Charles Peterson

Overview

👋 Welcome!

In this workshop, we will explore the advanced use of containers on HPC resources, like UCLA's Hoffman2. This is a follow-up to a previous workshop "Containers for HPC Resources"

- 🚀 Dive into advanced container topics
- 🧰 Build containers tailored for HPC resources
- 💡 Got suggestions for upcoming workshops?
- cpeterson@oarc.ucla.edu

📂 Files for this Presentation

👀 Viewing the slides

https://github.com/ucla-oarc-hpc/WS_MakingContainers

Find this slide deck in the `slides` folder.

- html format: https://ucla-oarc-hpc.github.io/WS_MakingContainers/
- PDF format: `WS_MakingContainers.qmd`

> **Note**
>
> This presentation was created with Quarto and RStudio.
>
> - Quarto file: `WS_MakingContainers.qmd`

Hands-on Exercise Setup

In this workshop, we will engage in hands-on exercises.

```
git clone https://github.com/ucla-oarc-hpc/WS_MakingContainers
```

Requirements:

- A computer with root/admin access (i.e., `sudo` permissions)
- Install Apptainer
- Install either Docker or Podman
  - In this workshop, we will be using Podman
- Note: Mac M1 (or other ARM-based systems) will not work for this exercise

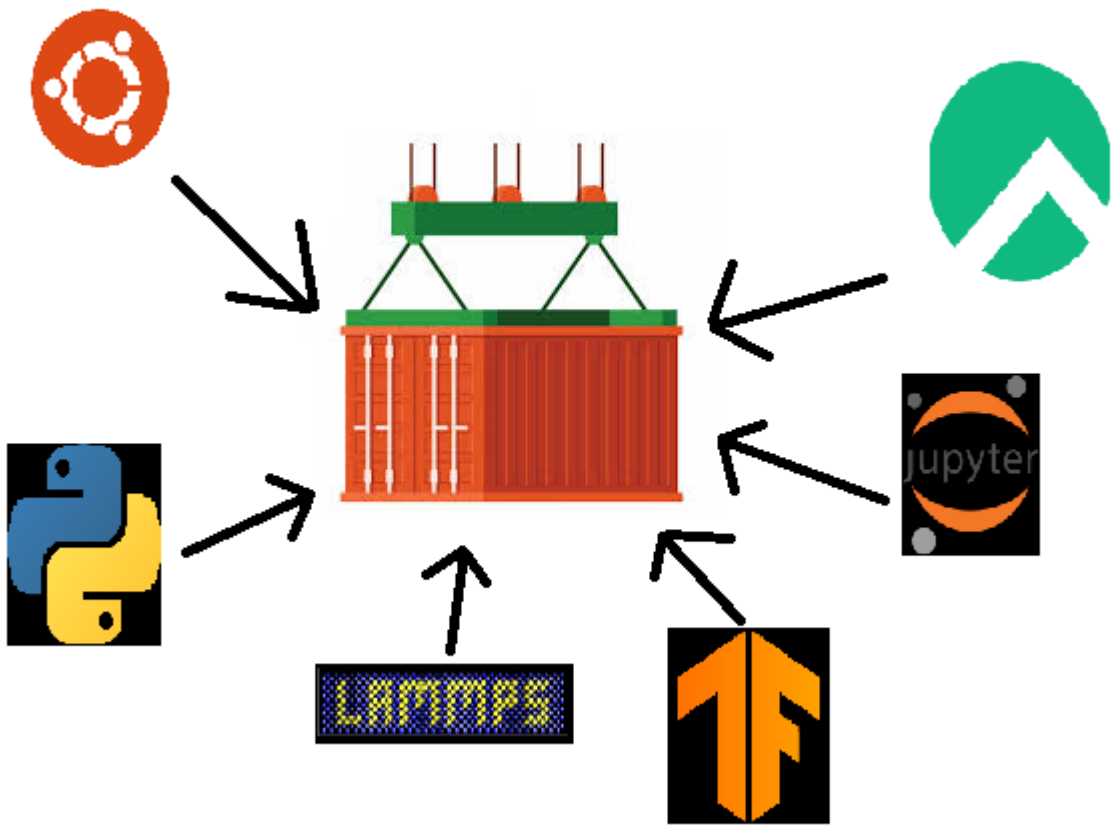
Alternative: Virtual Machine

Pre-configured Virtual Machine (VM) available on BOX

- VM file: `wscontainer.ova`
  - Recommended for use with VirtualBox
- Username/Password: wscontainer/wscontainer

Container Review

Containers



🛠️ Software for Containers

Apptainer

- Formerly Singularity
- 🎯 Designed and developed for HPC systems
- 🔧 Most likely installed on most HPC systems
- 🚀 Supports Infiniband, GPUs, MPI, and other devices on the Host
- 🐳 Can run Docker containers

🛠️ Software for Containers

Docker

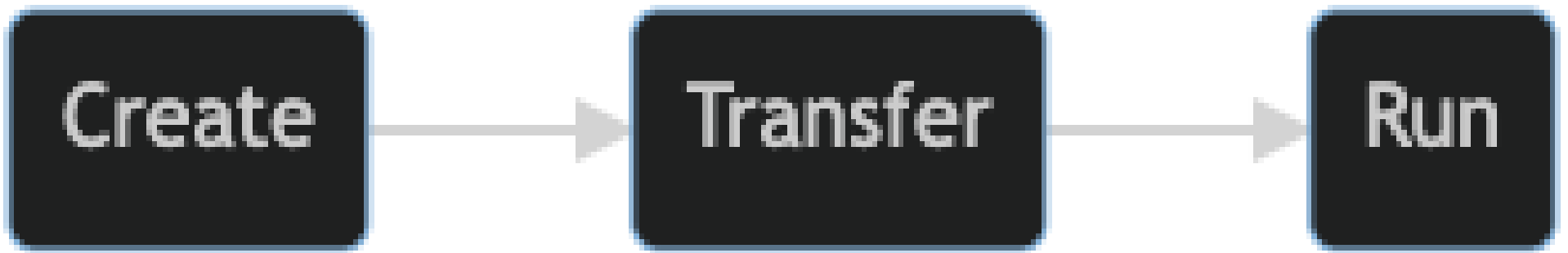
- ⭐ Very popular
- ☁️ Many popular cloud container registries
  - DockerHub, GitHub, Nvidia NGC
- 🚫 MPI not well supported
- 🚧 Most likely NOT available on many HPC systems

🔧 Software for Containers

Podman

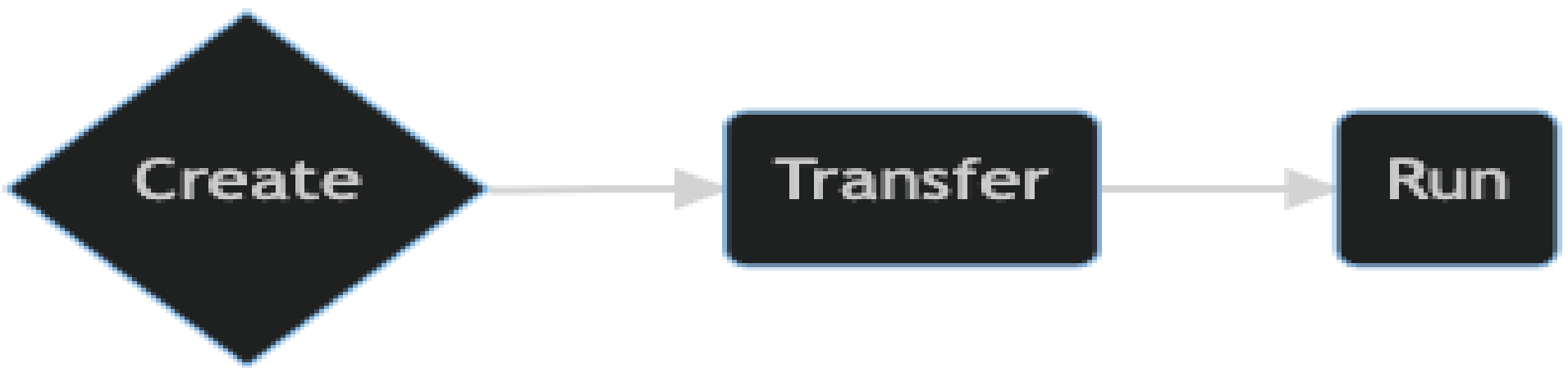
- 🔗 Similar syntax as with Docker
  - 🔄 Can use to ‘replace’ Docker
- 🚫 Doesn’t have a root daemon process
- 🖥️ On some HPC resources (not on Hoffman2, yet)

📋 Apptainer workflow



Apptainer workflow (Create)

- Build a container by installing Apptainer 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
- A SIF file contains the image for the container
- A sandbox container is a directory format used for writable containers



🔄 Apptainer workflow (Transfer)

🚚 Bring your container to Hoffman2

- 📩 Copy your container to Hoffman2

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

- 📩 Pull a container from online Container Register

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

- 🖥️ Use a container pre-built on Hoffman2

```
module load apptainer
ls $H2_CONTAINER_LOC
```

🏃 Apptainer workflow (Run)

🤖 Interactive (qrsh) session

```
qrsh -l h_data=15G
module load apptainer
apptainer exec mypython.sif python3 test.py
```

📄 Batch (qsub) job

```
cat << EOF >> myjob.job
module load apptainer
apptainer exec mypython.sif python3 test.py
EOF
qsub -l h_data=15G myjob.job
```

⭐ Common Usage

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

```
module load apptainer
```

- 🎉 Only module you need to load!
- No need to load tons of modules to run a single application
- 🌐 Expect MPI module if running parallel
  - `module load intel/2022.1.1`

📚 Common Apptainer commands

- 📥 Getting a container from somewhere

```
apptainer pull [options]
apptainer pull docker://ubuntu:22.04
```

- 🏭 Build a container

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

- 🏃 Run a single command inside of a container

```
apptainer exec [options]
apptainer exec myapp.sif MYCOMMAND
```

- 📜 Run the the container with a predefined script

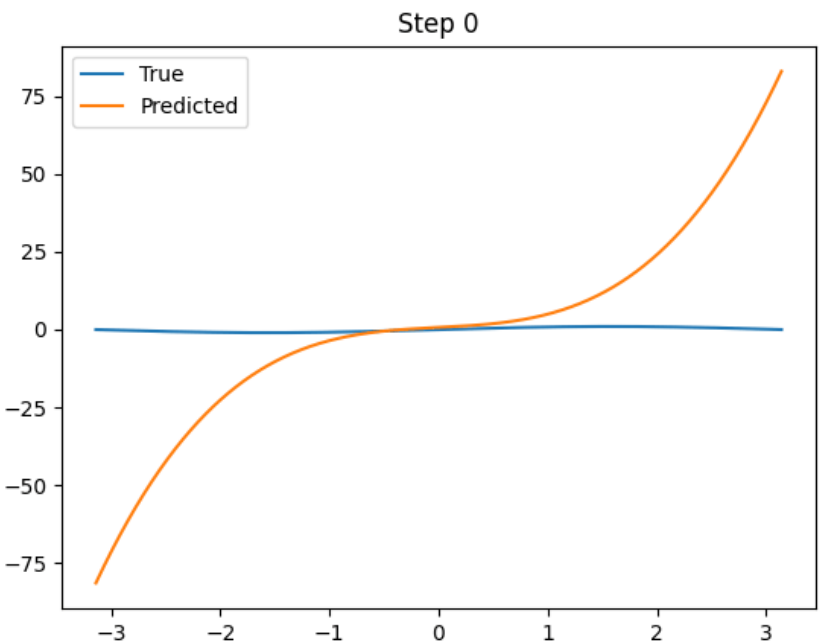
```
apptainer run [options]
apptainer run myapp.sif "SCRIPT ARGRMENTS"
```

Example 1: PyTorch

Example 1: Create writable containers 🛠️

- This example uses PyTorch 🧠
- Similar to last week's example
  - Though, instead of using PyTorch's pre-build container, we will create our own container

- Go to the EX1 directory
  - Examine the `pytorch.py` file
  - Optimize a 3rd order polynomial to a sine function



Creating the container 🚀

Sandbox image

- Create a container from a base source image
  - DockerHub has great minimal base containers
    - Ubuntu
    - Rocky Linux
    - CentOS
    - There are various versions of multiple builds

- Lets create a container based on Ubuntu 22.04

```
1  sudo apptainer build --sandbox ubuntu_22.04_SB/ docker://ubuntu:22.04
```

- sudo
  - Execute with root privileges
  - Building usually requires administrative access
- apptainer build:
  - Build new Apptainer container image
- –sandbox:
  - Flag for “sandbox” or a writable directory
- ubuntu_22.04_SB/
  - Name of the directory where the sandbox image will be created
- docker://ubuntu:22.04
  - Location of source image

Running the container 🏃

Next, we will start a WRITABLE interactive shell session in the sandbox image:

```
sudo apptainer shell --writable ubuntu_22.04_SB/
```

- `--writable` will allow us to modify the container

From here, we can run any commands we need to install PyTorch:

```
apt update
apt install -y python3 python3-pip
pip3 install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/
exit
```

Convert the sandbox container to a SIF file, `pytorch.sif`

```
sudo apptainer build pytorch.sif ubuntu_22.04_SB/
```

Running Our Container 🚀

Run our SIF container:

```
apptainer exec pytorch.sif python3 pytorch.py
```

We do not need to be root since we are just running python3.

Transfer our container:

```
scp pytorch.sif hoffman2.idre.ucla.edu:
```

Example job script to run on Hoffman2:

```
qsub pytorch.job
```

In this example, we used an interactive approach to install a python package in a container.

This is a useful way to experiment installing of your applications

Example 2: Building from definition files

Creating containers

I coded a chemistry app located on github.

- Very early attempt at learning Hartree-Fock
  - Named from my hedgie, Quill Ricker
- https://github.com/charliecpeterson/QUILL

To install, we need

- Python with the PySCF package
- Eigen3, a Linear Algebra library

- Instead of installing these dependencies on H2 (or loding modules)
- Lets build a container!!

We will build this container by:

- Apptainter definition file (.def)
- Using Docker/Podman (Dockerfile)

Apptainer Definition file

Definition Files are like the blueprint to building a custom container.

Instead of interactively modifying a sandbox image, we can build a container with this Definition file

The `quill.def` file has all steps needed to build the QUILL container.

```
Bootstrap: docker
From: ubuntu:20.04

%labels
Author Charles Peterson <cpeterson@oarc.ucla.edu>

%post
apt-get update
DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends
   git python3 python3-dev python3-pip \
   libeigen3-dev ca-certificates cmake make gcc g++
rm -rf /var/lib/apt/lists/*

pip3 install pyscf
ln -s /usr/bin/python3 /usr/bin/python
mkdir -pv /apps
cd /apps
git clone https://github.com/charliecpeterson/QUILL
cd QUILL
mkdir build ; cd build
cmake ..
make

%environment
export PATH=/apps/QUILL/build:$PATH
```

Sections:

- Bootstrap/From - Location of starting container
- `%labels` - adds metadata
- `%post` - This section runs commands to setup the final container
- `%environment` - define environment variables inside container

Create container

The `quill.sif` container is created

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

Let us test the container

- Run the command inside the container
  - `QUILL.x test.inp` inside the container

```
1  apptainer exec quill.sif  QUILL.x test.in
```

- Move container to Hoffman2

```
1  scp quill.sif H2USERNAME@hoffman2.idre.uc
```

Building using Docker/Podman

- Docker or Podman can create containers
  - Then we conver Docker/Podman container to Apptainer
- I will be using Podman
  - Docker and Podman have same syntax
  - Replace `podman` with `docker`

```
1 docker build
2 docker images
3 docker pull
4 docker run
```

```
1 podman build
2 podman images
3 podman pull
4 podman run
```

Dockerfile

- `Dockerfile-quill` file is used by Docker to create container

```
FROM ubuntu:20.04

## Author Charles Peterson <cpeterson@arc.ucla.edu>

RUN apt-get update \
      && DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends \
    git python3 python3-dev python3-pip \
    libeigen3-dev ca-certificates cmake make gcc g++ \
    && rm -rf /var/lib/apt/lists/*

RUN pip3 install pyscf ; ln -s /usr/bin/python3 /usr/bin/python

RUN mkdir -pv /apps \
     && cd /apps \
     && git clone https://github.com/charliecpeterson/QUILL \
     && cd QUILL \
     && mkdir build ; cd build \
     && cmake .. ; make

ENV PATH=/apps/QUILL/build:$PATH
```

Create container

Create Docker container

Build container from Dockerfile

```
podman build . -t quill:1.0 -f Dockerfile
```

See built docker (podman) container

```
podman image list
```

Create Apptainer container

Save docker (podman) image in tarball

```
podman save quill:1.0 > quill.tar
```

- Create SIF file

```
apptainer build quill.sif docker-archive:
```

- Transfer to Hoffman2

```
scp quill.sif H2USERNAME@hoffman2.idre.uc
```

Container Registry

In the previous slides, we created a SIF file (quill.sif), then transfer (scp) them to Hoffman2.

Instead of this, we can upload our container to a **Container Registry**.

- These **Registries** are used store our containers on a remote, cloud server that can then be pulled/download anywhere that has apptainer.
  - DockerHub
  - GitHub Packages

DockerHub

Lets create a repo on DockerHub

- First, create a DockerHub account
  - Mine is `charliecpeterson`
- Push our podman container to DockerHub
  - Registry location `docker.io`
- Tag container to DockerHub location

```
podman tag quill:1.0 docker.io/charliecpetersor
```

- Login info to DockerHub

```
podman login docker.io
```

- Push container to DockerHub

```
podman push docker.io/charliecpeterson/quill:1.
```

- Then pull the container on Hoffman2

```
apptainer pull docker://docker.io/charliecpeter
```

GitHub Packages

- Lets create a repo on GitHub
  - Look for the Packages tab
- Same syntax as before
  - registry location is `ghcr.io`

Push our final container to GitHub

```
podman tag quill:1.0 ghcr.io/charliecpeterson/quill:1.0
podman push ghcr.io/charliecpeterson/quill:1.0
```

Then pull the container on Hoffman2

```
apptainer pull docker://ghcr.io/charliecpeterson/quill:1.0
```

DockerHub and GitHub Packages are popular cloud registries. You can create and deploy a local container registry.

- https://docs.docker.com/registry/deploying/

Running Container

Once the container is on Hoffman2, submit job.

```
#!/bin/bash
#$ -cwd
#$ -o quill.$JOB_ID
#$ -j y
#$ -l h_rt=1:00:00,h_data=15G
#$ -pe shared 1
#$ -l arch=intel-gold*

# load the job environment:
. /u/local/Modules/default/init/modules.sh
module load apptainer

# Container part: apptainer exec QUILL.sif
# Command: QUILL.x /apps/QUILL/input.inp
time apptainer exec quill.sif QUILL.x test.inp
```

Submit job script

```
qsub test.job
```

More information on using Definition files

- https://apptainer.org/docs/user/1.0/definition_files.html

More information on using Dockerfiles

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

Example 3: Anaconda

Anaconda is a very popular python and R distributaion for simplifying package installation

- Check out my Hoffman2 Happy Hour on Anaconda on our GitHub workshop page

Though Anaconda can be tricky installing in a container due to environment setup.

We will have an example using Anaconda to install an application in a container.

Building H2O

We will go over creating a definition file for a example with Anaconda.

We will install the software h2o.ai. This is a great machine learning platform that has Python and R libraries.

In this example, we will use Anaconda to install h2o packages inside python and R.

H2o definition file

The `h2o.def` file

- Definition file for Apptainer
- Install Anaconda from an env.yml
- Use of `%runscript` to setup Anaconda env for `apptainer run`
  - the `$@` take arguments as a string from the command line
  - `apptainer run h2o.sif "python h2o-test.py"`

```
Bootstrap: docker
From: ubuntu:22.04


%labels
Author Charles Peterson <cpeterson@oarc.ucla.edu>


%post
export DEBIAN_FRONTEND=noninteractive
apt -y update ; apt -y upgrade
apt install -y  wget libbz2-dev wget git gcc  libreadline-dev zlib1g-dev default-jre default-jdk

#Install anaconda
cd /tmp
wget https://repo.anaconda.com/archive/Anaconda3-2022.05-Linux-x86_64.sh
```

```
bash Anaconda3-2022.05-Linux-x86_64.sh -b -p /opt/anaconda
bash -c "source /opt/anaconda/etc/profile.d/conda.sh
conda create -n h2oai h2o -c h2oai -c conda-forge
"

%runscript
exec bash -c "source /opt/anaconda/etc/profile.d/conda.sh
conda activate h2oai
$@"
```

Building container

- Create h2o.sif

```
sudo apptainer build h2o.sif h2o.def
```

- Run h2o.R inside the container

```
apptainer run h2o.sif "python h2o-test.py"
```

> **Note**
>
> - `apptainer exec foo.sif [COMMAND]`
>   - Run a single [COMMAND] inside the container
>   - The runscript will NOT run
> - `apptainer run foo.sif`
>   - Run the runscript inside the container

Example 4: RStudio and Jupyter

Jupyter

- Create a container with Jupyter
  - Useful so you can run same Jupyter setup anywhere
  - Install all packages you need in container
- `jupyter.def`
  - Apptainer Def file
  - Starts with a pre-built python 3.8.13 container
  - Adds jupyter, pandas, and seaborn

- Build container (locally)
- Transfer to H2

```
sudo apptainer build jupyter.sif jupyter.def
scp jupyter.sif hoffman2.idre.ucla.edu:
```

- Start Jupyter on Hoffman2
  - Note the name of compute node you landed on

```
qrsh -l h_data=10G
module load apptainer
hostname
apptainer exec jupyter.sif jupyter lab --ip 0.0.0.0
```

- SSH tunnel to H2 compute node
  - Change nXXX (compute node)
  - Change port 8888 if needed

```
ssh  -L 8888:nXXX:8888 username@hoffman2.idre.ucla.edu
```

- Then open a web browser and type

```
http://localhost:8888
```

Rstudio

- Use Rstudio Server to open a Rstudio session on your web browser.
  - Very similar to the Jupyter setup
- More information from a previous H2HH Rstudio session
- Create this container using Docker/Podman
  - Local computer

```
podman build -f Dockerfile-rstudio -t rstudio:4.1.0 .
podman save rstudio:4.1.0 > rstudio.tar
apptainer build rstudio.sif docker-archive://rstudio.ta
scp rstudio.sif hoffman2.idre.ucla.edu
```

- Setup Rstudio on Hoffman2

```
# get an interactive job
qrsh -l h_data=10G
# Create tmp directories
mkdir -pv $SCRATCH/rstudiotmp/var/lib
mkdir -pv $SCRATCH/rstudiotmp/var/run
mkdir -pv $SCRATCH/rstudiotmp/tmp
#Setup apptainer
module load apptainer
#Run rstudio
apptainer run -B $SCRATCH/rstudiotmp/var/lib:/var/lib/r
# This command will display some information and a `ssh
```

- Start Rstudio on web browser

```
ssh  -L 8787:nXXX:8787 username@hoffman2.idre.ucla.edu
# Then open a web browser and type
http://localhost:8787 #or whatever port number that was
```

Things to Think About 🤔

Tips 🛠️

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
- Check out `Multi-Stage` approach

- Build from an existing container
  - Look for images on DockerHub, NGC
  - Build your own and upload to DockerHub/GitHub

- Good idea to build a sandbox container, then create definition file
  - Test out commands in sandbox while making the def file!

More Things to Think About 🧠

- Share .sif files with your friends!
  - Save your (Docker) containers to DockerHub or GitHub Packages
  - Create Dockerfile/Def files to recreate or modify containers
- Find examples of Dockerfiles and Apptainer def files on our GitHub Page
  - https://github.com/ucla-oarc-hpc/hpc_containers
- Use DMTCP Checkpointing with Containers!!
  - https://apptainer.org/docs/user/main/checkpoint.html

Thank you! 🙏

Questions? Comments? 🤔

Charles Peterson cpeterson@oarc.ucla.edu