REPRODUCIBLE PIPELINES IN HPC WITH APPTAINER

Jerry Li Ph.D.

Research Support Analyst

Digital Research Services, IST

jiarui.li@ualberta.ca

UNIVERSITY OF ALBERTA

Outline

1. Introduction to Apptainer container
2. Use a pre-built Apptainer container
3. Make a custom Apptainer container
4. Applications of Apptainer container
5. Q&A

UNIVERSITY OF ALBERTA

Objectives

- Learn what containers are

- Understand when to use Apptainer containers on HPC

- Know how to use Apptainer container on HPC systems

- Able to build custom Apptainer container

Note

- The slides can be found in Github: https://github.com/ualberta-rcg/Apptainer_Container_On_HPC

- There are also some useful links below:
  - Apptainer home
  - **Apptainer Documentation**
  - Apptainer on GitHub

  - Singularity Hub
  - **Docker Hub**

- Please reach out if you have any questions about the documentation or would like to see any additions.

UNIVERSITY OF ALBERTA

1. Introduction to Apptainer Container

HPC Software Pain Points

- Difficult to be installed in HPC system by a user:
    - Dependencies are not available in the host system (e.g. HPC cluster GLIBC version is too low)
    - Software installation needs admin power such as "yum install", "apt-get"
    - Cannot use Conda in many HPC clusters (e.g. National systems such as Cedar)

- Difficult to share tools and/or workflows with others

- Reproducibility is not guaranteed (e.g. new software stack installed in the system).

UNIVERSITY OF ALBERTA
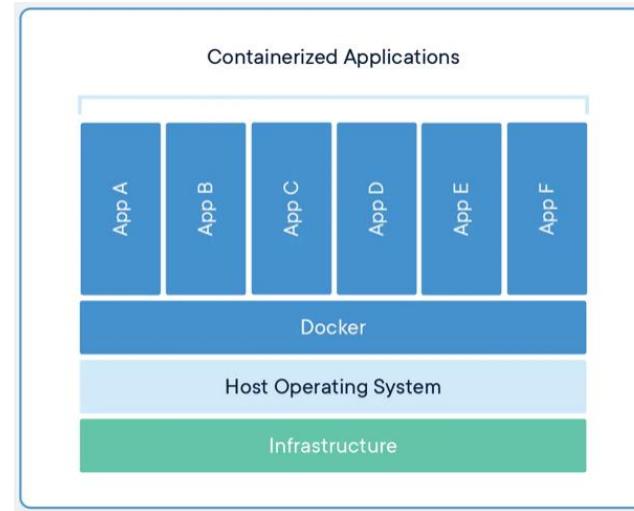
How can these pain points be addressed?

Apptainer containers! (Previously called Singularity)

- Designed for HPC

- It assumes you don't have root access when using it (*not building it)

- Easy to share and reproduce

- Independent of the host environment
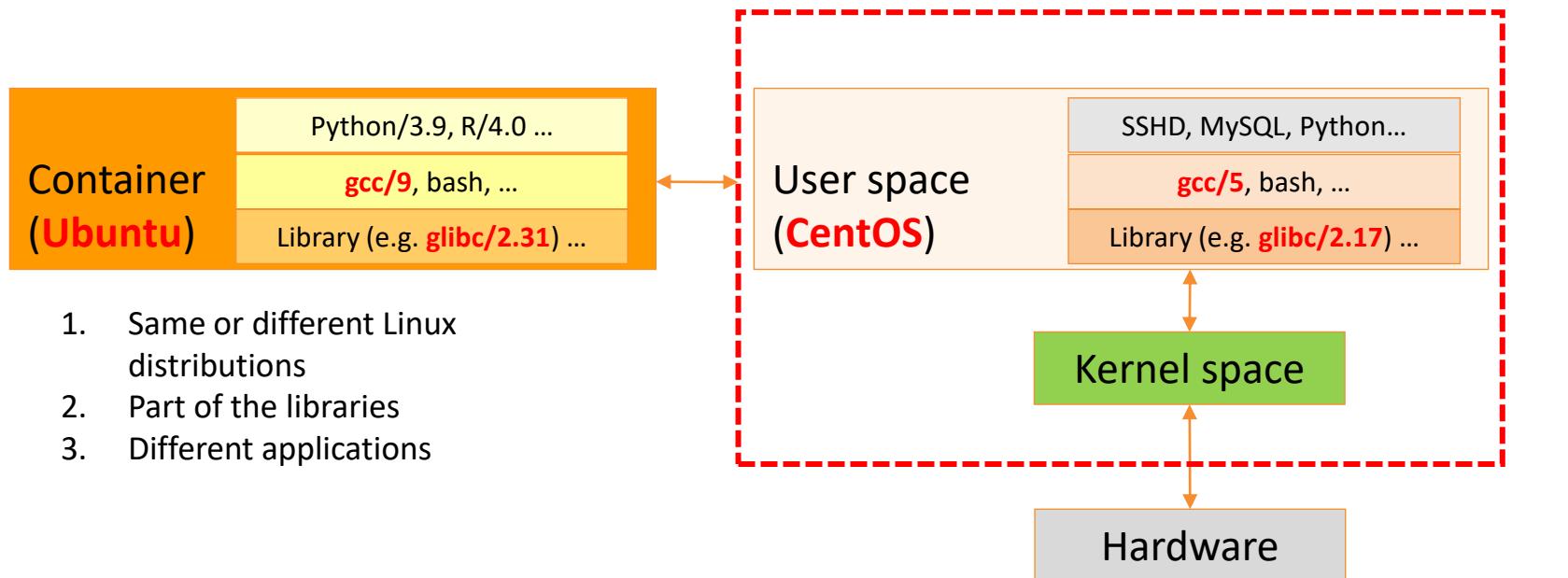
What is a container?

- Package of code, dependencies, and libraries necessary to run software in (nearly) any computing environment

- Provides virtualization at the operating system level

- Two main types of containers you may have heard of: Docker and Apptainer (singularity)

- Containers are stored as image files
  - Apptainer = .sif (Singularity Image Format)
  - Dockerfile = no extension

Source: https://www.docker.com/resources/what-container/

UNIVERSITY OF ALBERTA

What is a container?

Container (**Ubuntu**)	Python/3.9, R/4.0 ...
	gcc/9, bash, ...
	Library (e.g. **glibc/2.31**) ...

1. Same or different Linux distributions
2. Part of the libraries
3. Different applications

User space (**CentOS**)	SSHD, MySQL, Python...
	gcc/5, bash, ...
	Library (e.g. **glibc/2.17**) ...

Kernel space

Hardware

Note: Singularity rebranded as "Apptainer"

https://apptainer.org/

Singularity joined Linux Foundation November 2021.

Sockeye and Cedar have singularity 3.3-3.8 for now.

Singularity is using "Apptainer" since version 3.9, and we will eventually use "Apptainer" as the executable instead of singularity.

Question?

UNIVERSITY
OF ALBERTA

2. Use pre-built container

Get an account and login

-

Pull down the container image from dockerhub

- Pull down a container image, examples:

```
mkdir apptainer
cd apptainer
salloc --time=4:00:00 --account=def-sponsor00 --cpus-per-task=1 --mem=2G
module load apptainer
apptainer pull docker://python:3.11.13
```

Repository space of pre-built container

- Find pre-built container images:
  - **Docker Hub: https://hub.docker.com/**

  - Singularity Hub: https://singularityhub.github.io/singularityhub-docs/
    Read-only and not maintained

  - NVIDIA GPU Cloud (NGC) Catalog for AI, HPC, and Visualization: https://docs.nvidia.com/ngc/ngc-catalog-user-guide/index.html

UNIVERSITY OF ALBERTA

Pull down a different version

- Check the tag in dockerhub and specify the version by ":"

```
module load apptainer
apptainer pull docker://python:3.8
```

Use the container

- Two ways of running the program in a container:

```
apptainer shell python_3.11.13.sif
python --version
exit
```

```
apptainer exec python_3.11.13.sif python --version
```

Your program is inside the container

```
apptainer exec python_3.11.13.sif python --version
python --version
```

SIF is not editable

- ```
Outside the container:
ls -l /usr/local/bin
mkdir /usr/local/bin/test

Inside the container:
apptainer exec python_3.11.13.sif ls -l /usr/local/bin
apptainer exec python_3.11.13.sif mkdir /usr/local/bin/test
```

Everything is inside .sif                    .sif is not editable
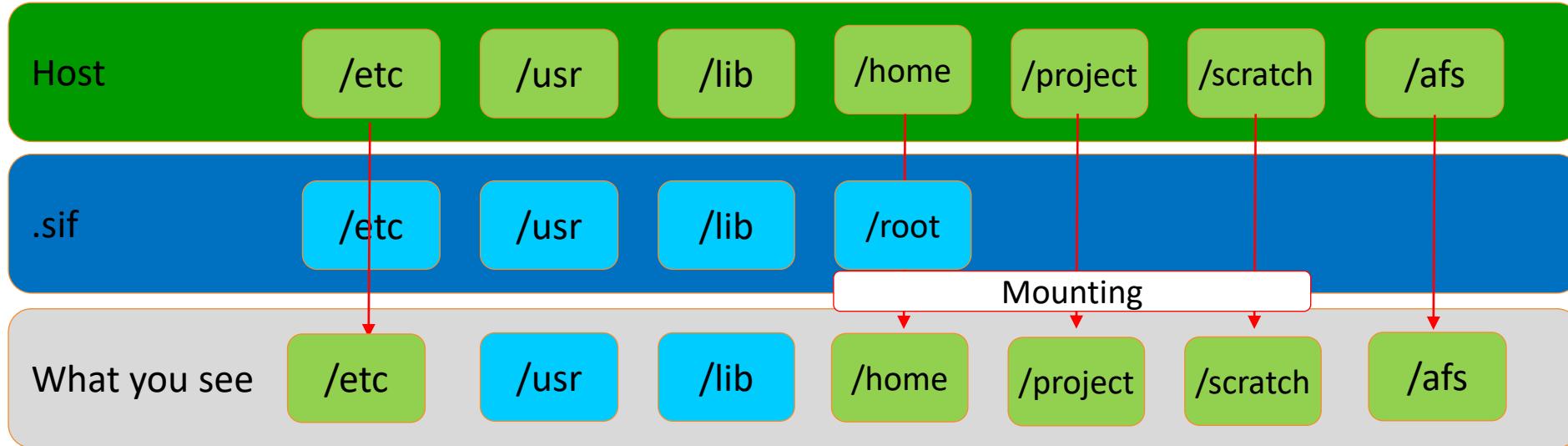
How can I edit my
input/output?

# Use the input/output in the host

- Read input in the host

```
echo -e "import sys\nprint(sys.version)" > $HOME/test.py
python $HOME/test.py
apptainer exec python_3.11.13.sif python $HOME/test.py
```

File system of Apptainer

File system of Apptainer

- Compare the root directory in the host with Apptainer container

In host:

```
ls -l /
ls -ld /afs
ls -l $HOME
echo $HOME
```

- In container

```
apptainer exec python_3.11.13.sif ls -l /
apptainer exec python_3.11.13.sif ls -ld /afs
apptainer exec python_3.11.13.sif ls -l $HOME
apptainer exec python_3.11.13.sif echo $HOME
```

IMPORTANT: clean the cache often

- Check the cache space

```
cd $HOME
ls -la
cd .apptainer
du -h


apptainer cache clean
du -h
```

Practice: fill the following job script

- How to run apptainer in a batch job

```bash
#!/bin/bash
#SBATCH --account=def-sponsor00
#SBATCH --time=0-0:05:00
#SBATCH --cpus-per-task=1
#SBATCH --ntasks=1
#SBATCH --nodes=1
#SBATCH --mem=2G

module load apptainter
apptainer exec …
```

- Two ways of running the program in a container:

```
apptainer shell python_3.11.13.sif
python --version
exit
```

```
apptainer exec python_3.11.13.sif python --version
```

Question?

UNIVERSITY
OF ALBERTA

3. Make a custom container

Note: run the following in another window

- Open two SSH session

```
Session1:Sandbox
Session2:SIF
```

- Run the following command on Session1

```
salloc --time=4:00:00 --account=def-sponsor00 --cpus-per-task=1 --mem=2G
module load apptainer
mkdir apptainer && cd apptainer
apptainer build --sandbox --fakeroot python_3.11.13.sandbox docker://python:3.11.13
```

- Run the following on Session2

```
module load apptainer
cd apptainer
apptainer pull docker://python:3.11.13
```

Run this command first

- Session1: Sandbox

```
# Sandbox
APPTAINER_BIND= apptainer shell --writable --fakeroot -c -e python_3.11.13.sandbox/
```

- Install the dependencies

```
# Sandbox
apt-get update && apt-get install gdal-bin libgdal-dev
```

Why do we want to make a custom container

- I want to install a tool/package/module

- I want to change the environment (i.e. activate a virtual environment automatically)

- I want to use Conda, which is not supported by the Alliance cluster

- I want to build my pipeline into the container and share it with others

UNIVERSITY OF ALBERTA

Install a python package

- Install the python package by pip:

```
#Session2: SIF
cd $HOME/apptainer
apptainer shell python_3.11.13.sif
export REQUESTS_CA_BUNDLE=/etc/ssl/certs/ca-certificates.crt
pip install emoji
```

```
python
import emoji
print(emoji.emojize("Python is fun :snake:"))
quit()
```

Install a python package

- You may encounter errors:

```
#SIF
pip install GDAL==3.6
```

Install a python package

- Try with sandbox

- Install the dependencies

```
# Sandbox
APPTAINER_BIND= apptainer shell --writable --fakeroot -c -e python_3.11.13.sandbox/
apt-get update && apt-get install gdal-bin libgdal-dev          # already run
pip install emoji
pip install GDAL==3.6
python -c "from osgeo import gdal"
```

Where are those packages installed

- Install packages in sandbox
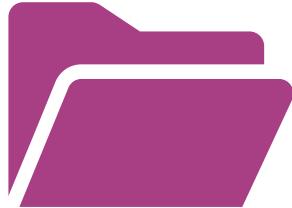
```
# Sandbox
pip show GDAL
pip show emoji
exit
```

Convert Sandbox To .sif

```
# Sandbox
exit
apptainer build python_GDAL.sif python_3.11.13.sandbox
```

Sandbox ⇔ SIF

Sandbox

Singularity Image File (SIF)

convertible

- A directory
- Editable
- Not easy to share

- A single file
- NOT editable
- Easy to share

apptainer build <output> <input>

UNIVERSITY
OF ALBERTA
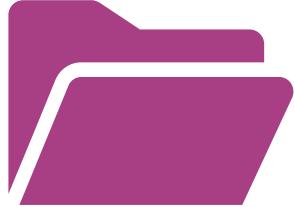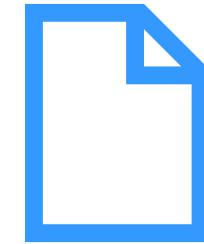
Question?

4. Applications of Apptainer Container

Sandbox ⇔ SIF

Sandbox   Singularity Image File (SIF)

convertible

- A directory   - A single file
- Editable   - NOT editable
- Not easy to share   - Easy to share

Question:
Can I delete Sandbox after generating .sif?

.sif ⇔ sandbox

```
cd $HOME/apptainer
ls -l
```

```
python_3.11.13.sandbox
python_3.11.13.sif
python_GDAL.sif
```

UNIVERSITY OF ALBERTA
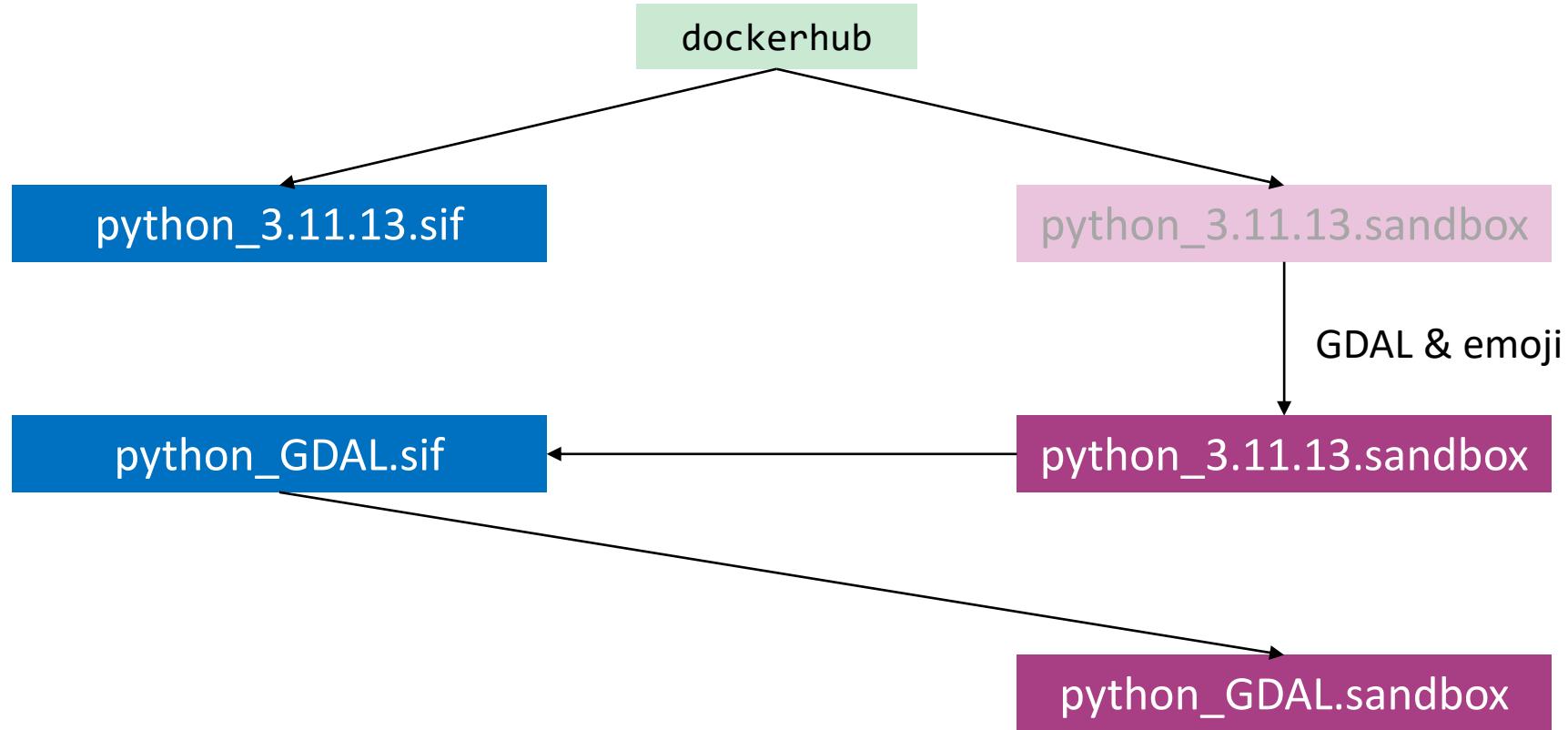
.sif ⇔ sandbox

```
apptainer build python_GDAL.sandbox python_GDAL.sif
```

```
Question: what's the difference between python_GDAL.sandbox and python_3.11.13.sandbox ?

A) python version
B) Whether emoji is installed
C) Whether GDAL is installed
D) No difference
```

.sif ⇔ sandbox

Install Conda inside the container

```
APPTAINER_BIND= apptainer shell --writable --fakeroot -c -e python_3.11.13.sandbox/
mkdir /conda        # already run
cd /conda
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O miniconda.sh
bash miniconda.sh -b -u -p /usr/local/miniconda3
rm miniconda.sh
source /usr/local/miniconda3/bin/activate
conda create -n test
conda activate test
conda install bioconda::bwa
echo 'source /usr/local/miniconda3/bin/activate test' >> /environment
cat /environment
exit
apptainer exec python_3.11.13.sandbox bwa
```

Definition file (.def)

```
Bootstrap: docker
From: python:3.11.13
Stage: build
%environment
        export LC_ALL=C
        source /usr/local/miniconda3/bin/activate test
%post

        apt update && apt install -y wget
        mkdir /conda
        cd /conda
        wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O miniconda.sh
        bash miniconda.sh -b -u -p /usr/local/miniconda3
        rm miniconda.sh
        export PATH="/usr/local/miniconda3/bin":$PATH
        eval "$(/usr/local/miniconda3/bin/conda shell.bash hook)"
        conda tos accept --override-channels --channel https://repo.anaconda.com/pkgs/main
        conda tos accept --override-channels --channel https://repo.anaconda.com/pkgs/r
        conda create -y -n test
        conda activate test
        conda install -n test -y bioconda::bwa
        rm -rf /conda
%runscript

%startscript

%test

%labels

%help
```

```
cp /project/def-sponsor00/apptainer/conda.def .
apptainer build conda.sif conda.def
```

Definition file (.def)

```
Bootstrap: localimage
From: conda.sif
Stage: build

%environment

%post
    conda install -c bioconda samtools "openssl=1.0"

%runscript

%startscript

%test

%labels
    UofA Bootcamp
    Date 2025-10-07

%help
    This is a container for training
```

```
cp /project/def-sponsor00/apptainer/samtools.def .
apptainer build samtools.sif samtools.def
apptainer exec samtools.sif samtools --help
```

Rstudio with Apptainer

```
# Get rstudio_latest.sif from dockerhub: https://hub.docker.com/r/rocker/rstudio
sbatch rstudio.sh
```

```
Practice: What Apptainer function should I use to get rstudio_latest.sif from dockerhub?

A) apptainer shell
B) apptainer pull
C) apptainer exec
D) apptainer create
```

File System − Mount Host Directory

```
/cvmfs
apptainer exec python_3.11.13.sif ls /cvmfs
apptainer exec -B /cvmfs python_3.11.13.sif ls /cvmfs
apptainer exec -B /cvmfs:/software python_3.11.13.sif ls /software
```

File System – Reset tmp and var_tmp

```
mkdir tmp
yes "abcdef" | head -n 5000000 > big.txt

apptainer exec python_3.11.13.sif timeout --signal=KILL 1s sort --buffer-size=256K $PWD/big.txt > /dev/null || true
ls /tmp
ls tmp


apptainer exec -W $HOME/apptainer/tmp -c -B $PWD python_3.11.13.sif timeout --signal=KILL 1s sort --buffer-size=256K $PWD/big.txt > /dev/null || true
ls tmp


# For heavy I/O
mkdir $SLURM_TMPDIR/outputs
apptainer exec -W $SLURM_TMPDIR/outputs -c -B $PWD python_3.11.13.sif timeout --signal=KILL 1s sort --buffer-size=256K $PWD/big.txt > /dev/null || true
ls $SLURM_TMPDIR/outputs/tmp
```

Running MPI Job with Apptainer

```
Bootstrap: docker
From: mfisherman/openmpi:4.1.5
Stage: build
%environment

%files

        hello_mpi.c             /usr/local/bin/hello_mpi.c

%post

        mpicc -o /usr/local/bin/hello_mpi /usr/local/bin/hello_mpi.c
```

```
cd $HOME/apptainer
mkdir MPI
cd MPI
cp /project/def-sponsor00/apptainer/hello_mpi.* .
module load apptainer
apptainer build --fakeroot --ignore-fakeroot-command openmpi.sif hello_mpi.def
sbatch hello_mpi.sh


# Why do we use openmpi/4.1.5?
mpirun --version
```

Use GPU

- Pull down a pytorch container

```
apptainer pull docker://pytorch/pytorch:2.10.0-cuda13.0-cudnn9-runtime
```
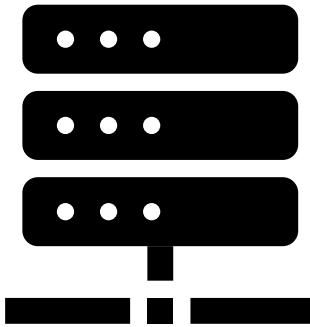
- Prepare your python script

```
import torch;
print('cuda:', torch.cuda.is_available());
x=torch.randn(4096,4096,device='cuda');
y=x@x; print('ok', y[0,0].item())
```
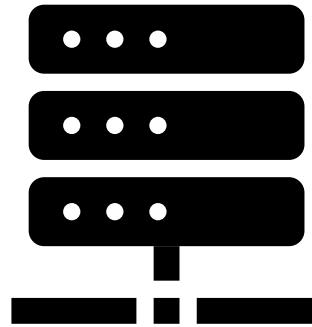
- Add --nv when running GPU jobs

```
apptainer exec --nv pytorch_2.10.0-cuda13.0-cudnn9-runtime.sif python
test_pytorch.py
```

UNIVERSITY OF ALBERTA

Apptainer is good for sharing

Rocky9
Python/3.12
R/4.5

Ubuntu:
Python/3.10
R

Question?

UNIVERSITY
OF ALBERTA

Thank you!