

# REPRODUCIBLE PIPELINES IN HPC WITH APPTAINER

**Jerry Li Ph.D.**

Research Support Analyst

Digital Research Services, IST

[jiarui.li@ualberta.ca](mailto:jiarui.li@ualberta.ca)

October 7, 2025



**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

# 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](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.

# 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).

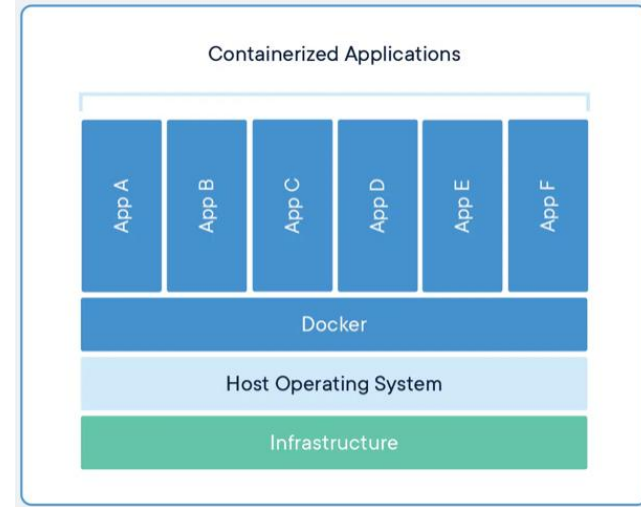
# How can these pain points be addressed?

## 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?

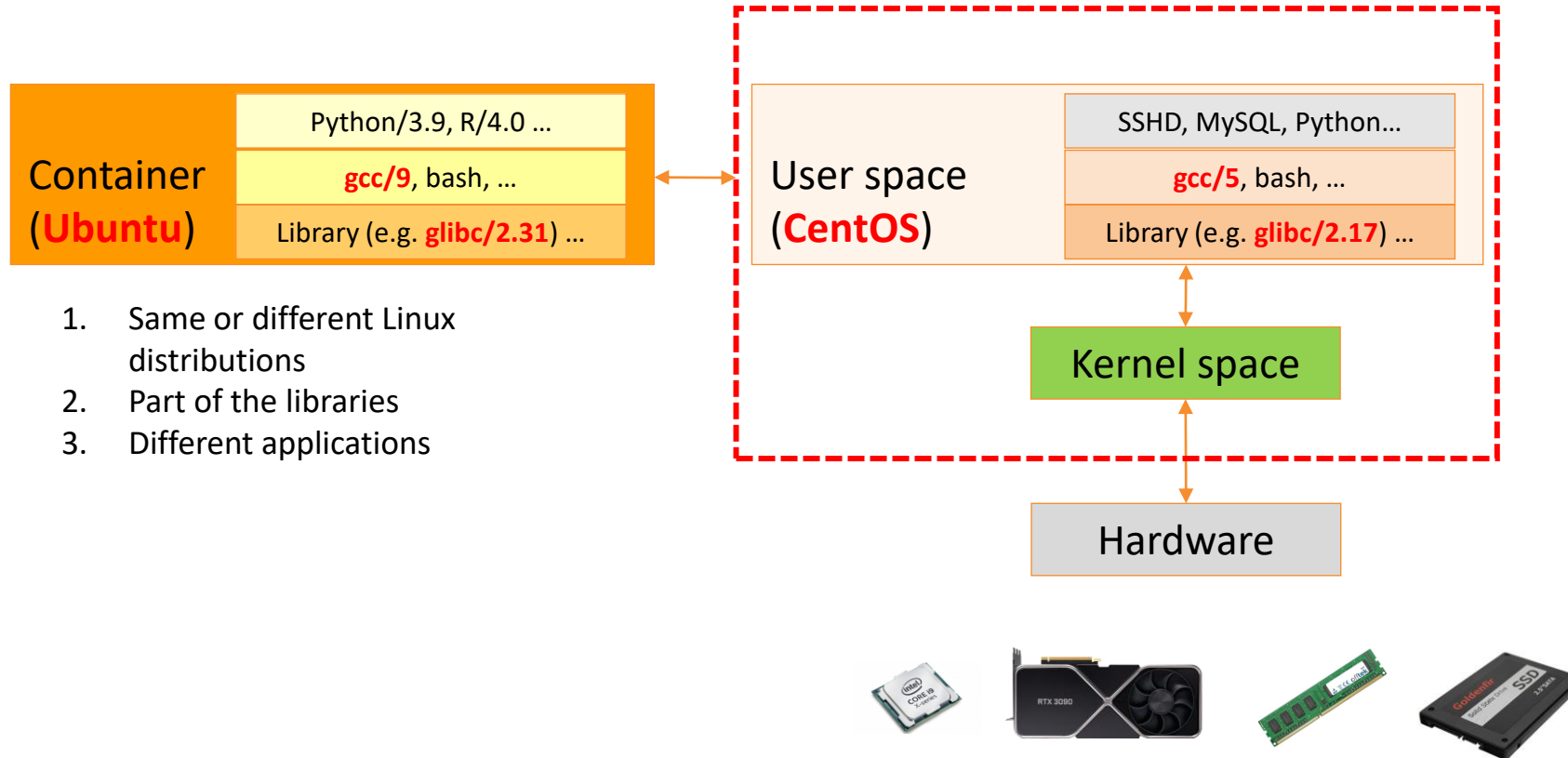
- Package of code, dependencies, and libraries necessary to run software in (nearly) any computing environment
- Provides virtualization at the operating system level
- Two main types of containers you may have heard of: Docker and Apptainer (singularity)
- Containers are stored as image files
  - Apptainer = .sif (Singularity Image Format)
  - Dockerfile = no extension



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



# What is a container?



# 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?

## 2. Use pre-built container



# Get an account and login

- Open the doc:

<https://tinyurl.com/DRS-Apptainer>

- Login:

```
ssh user00@fall2025-uofa.c3.ca  
Password: eltoroloco-abc123
```

# Pull down the container image from dockerhub

- Pull down a docker 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>

# 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
```

# 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
```

# 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

- Compare the root directory in the host with Apptainer container

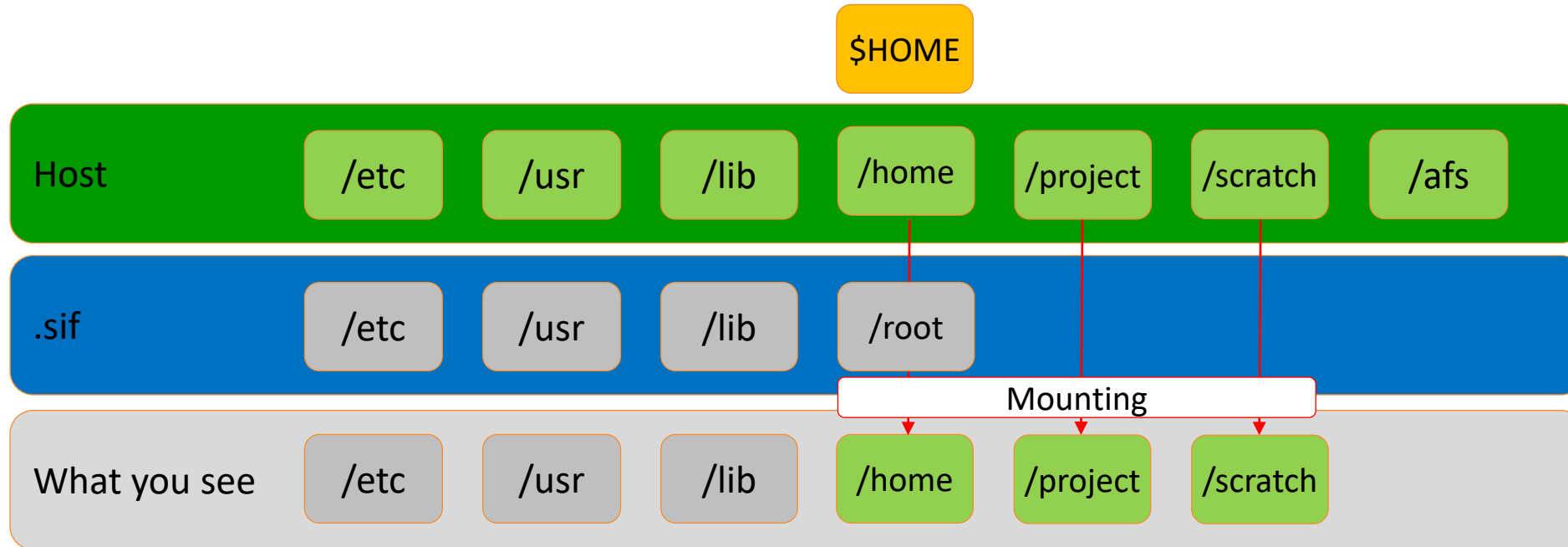
In host:

```
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
```

# File system of Apptainer



# 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

```
#!/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
#SBATCH -o test.out
#SBATCH -e test.err

<command>
```

- 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?



# 3. Make a custom container



# Note: run the following in another window

- Start a new SSH session

```
# Sandbox
```

- Run the following command

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

# Run this command first

- Open your second ssh window with sandbox created:

```
# 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

# Install a python package

- Install the python package by pip:

```
# SIF
```

```
cd $HOME/apptainer
```

```
apptainer shell python_3.11.13.sif
```

```
cp /etc/ssl/certs/ca-certificates.crt ~/my-ca-bundle.crt
```

```
export REQUESTS_CA_BUNDLE=~/my-ca-bundle.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.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
```

```
apptainer build python_new01.sif python_3.11.13.sandbox  
exit
```

# Sandbox ↔ SIF

Sandbox



- A directory
- Editable
- Not easy to share

Singularity Image File (SIF)



- A single file
- NOT editable
- Easy to **share**

convertible



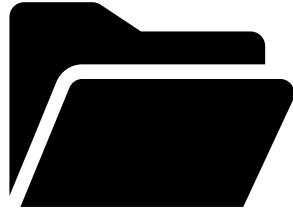
```
apptainer build <output> <input>
```

# Question?

# 4. Applications of Apptainer Container

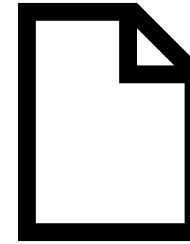
# Sandbox ⇔ SIF

Sandbox



- A directory
- Editable
- Not easy to share

Singularity Image File (SIF)



convertible



- A single file
- NOT editable
- Easy to **share**

Question:

Should I delete Sandbox after generating .sif?

# .sif ↔ sandbox

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

```
python_3.11.13.sandbox  
python_3.11.13.sif  
python_new01.sif
```

```
# if you don't have these files/folders, please do:  
mkdir /apptainer  
cp -r /project/apptainer/day1/* apptainer/
```

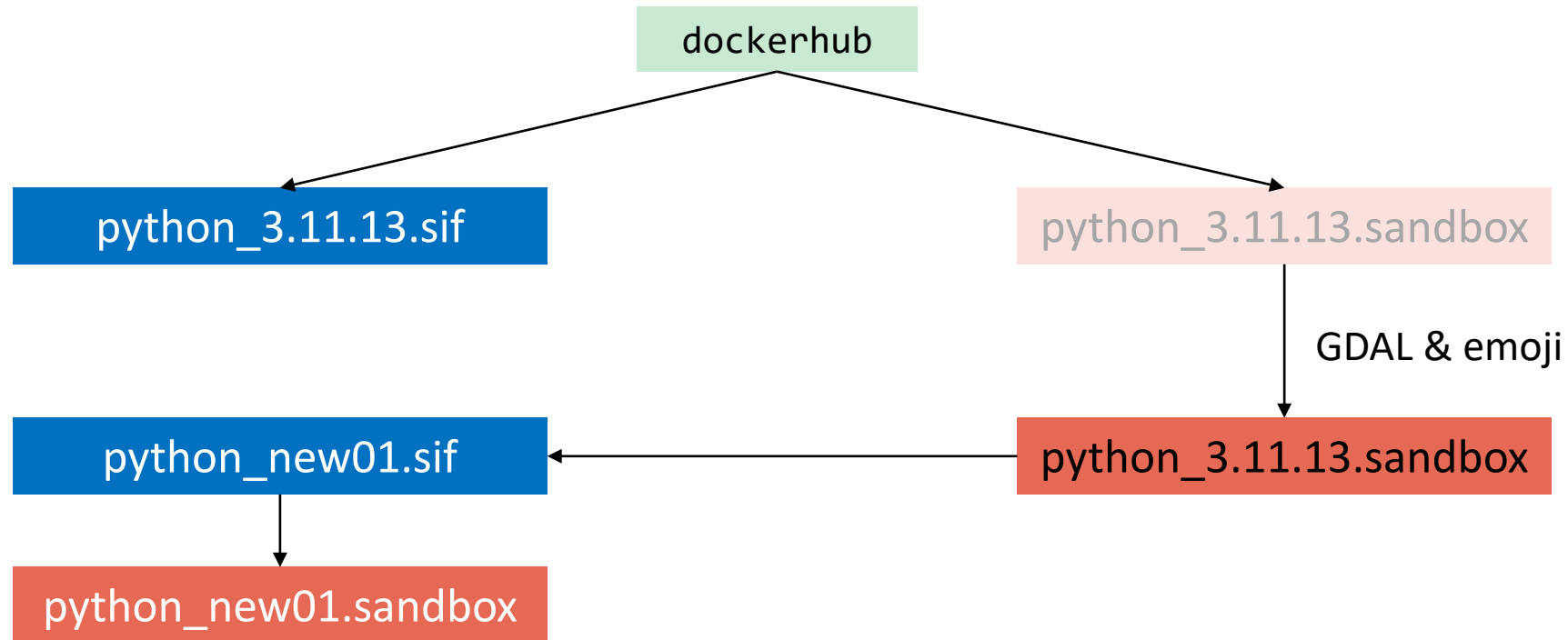
# .sif ↔ sandbox

```
apptainer build python_new01.sandbox python_new01.sif
```

Question: what's the difference between `python_new01.sandbox` and `python_3.11.13.sandbox` ?

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

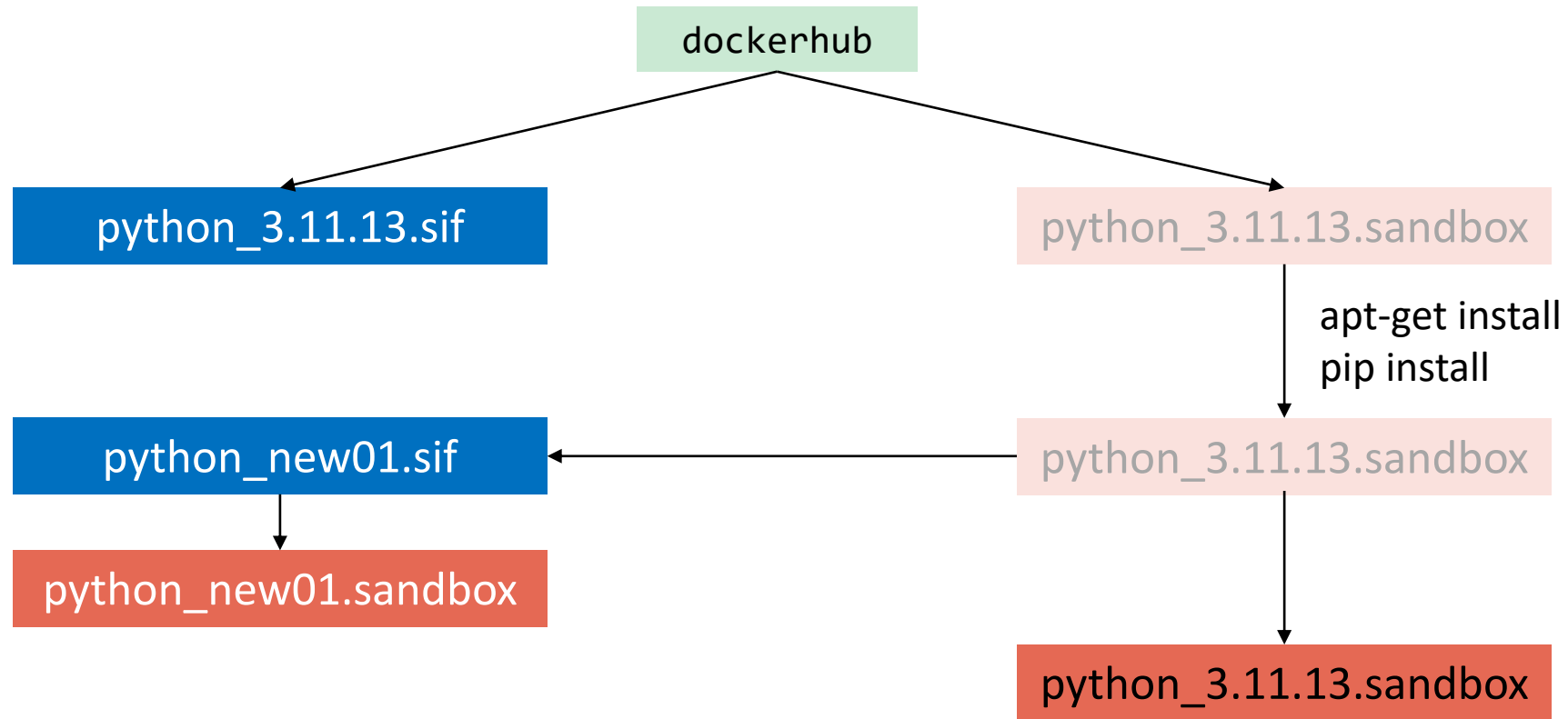
# .sif ⇔ sandbox



```
salloc --time=4:00:00 --account=def-sponsor00 --cpus-per-task=1 --mem=2G  
module load apptainer  
APPTAINER_BIND= apptainer shell --writable --fakeroot -c -e python_3.11.13.sandbox/  
mkdir /conda
```



# .sif ⇔ sandbox



```
exit  
rm -rf python_new01.*
```

# 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 miniconda3  
rm miniconda.sh  
source miniconda3/bin/activate  
conda create -n test  
conda activate test  
conda install bioconda::bwa  
echo 'source /conda/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 create -y -n test
    conda activate test
    conda install -n test -y bioconda::bwa
    rm -rf /conda
%runscript
%startscript
%test
%labels
%help
```

# Definition file (.def)

```
Bootstrap: localimage
From: /home/user070/apptainer/example.1.sif
Stage: build

%environment

%post
    conda install -n test -y bioconda::samtools

%runscript

%startscript

%test

%labels
    UofA Bootcamp
    Date 2025-10-07

%help
    This is a container for training
```

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

# Jupyter Notebook with Apptainer

```
cp /project/def-sponsor00/apptainer/jupyter.sh .  
sbatch jupyter.sh
```

Practice: What Apptainer function should I use to get datascience-notebook.sif from dockerhub?

- A) apptainer shell
- B) apptainer pull
- C) apptainer exec
- D) apptainer push

# File System – Mount Host Directory

```
mkdir hello
```

```
echo 'hello world' > hello/hello_world.txt
```

```
apptainer exec python_3.11.13.sif cat /hello/hello_world.txt
```

```
apptainer exec -B $PWD/hello:/hello python_3.11.13.sif cat /hello/hello_world.txt
```

```
apptainer exec -B $PWD/hello:/usr/world python_3.11.13.sif cat /usr/world/hello_world.txt
```

# 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
```

```
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
```

# 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
```

# 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/MPI/hello_mpi.* .
module load apptainer
apptainer build --fakeroot --ignore-fakeroot-command openmpi.sif hello_mpi.def # you can skip
this step as the .sif is already built

# Why do we use openmpi/4.1.5?

sbatch hello_mpi.sh
```



# Use GPU

- Add `--nv` when running GPU jobs

```
apptainer exec --nv container.sif python model.py
```

# Question?

# Thank you!