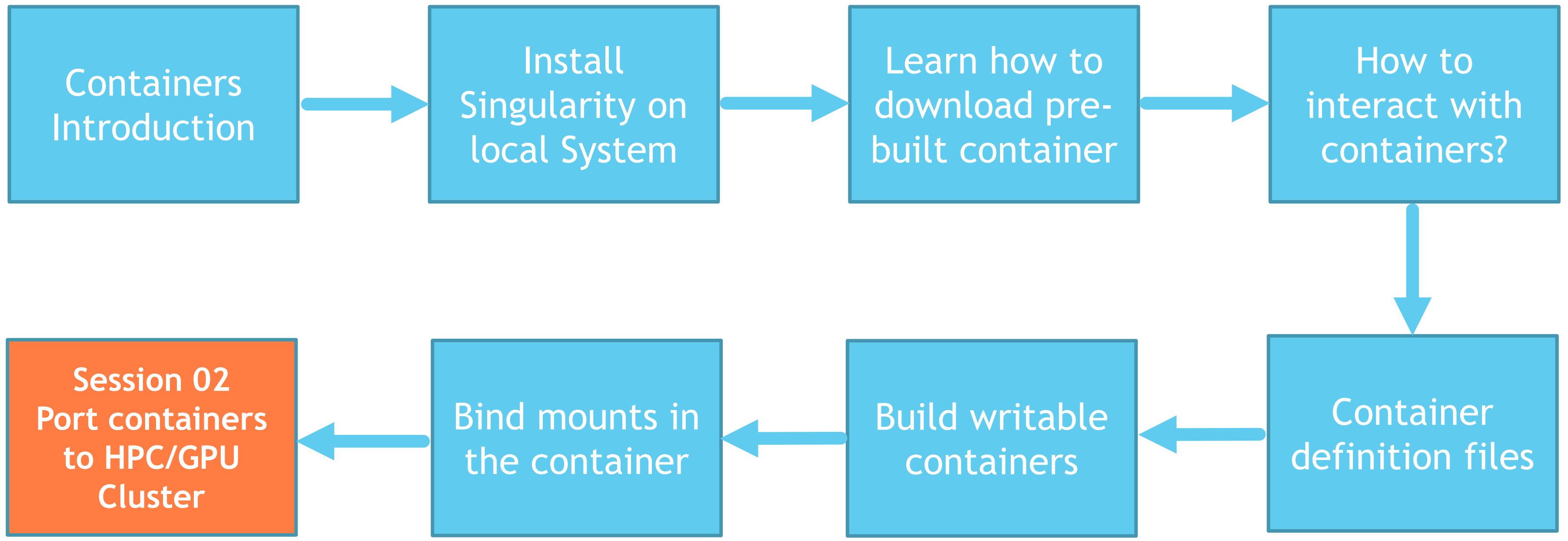
INTRODUCTION TO LINUX CONTAINERS SESSION 01

Mustafa Arif
mustafa.arif@qatar.tamu.edu
Texas A&M University at Qatar

Session 01: What we are going to learn today?


Containers Introduction
Install Singularity on local System
Learn how to download pre-built container
How to interact with containers?
Container definition files
Build writable containers
Bind mounts in the container
Session 02 Port containers to HPC/GPU Cluster

Documentation Link

https://rc-docs.qatar.tamu.edu/index.php/Linux_containers

Training guidelines

Each attendee must have received login credentials for a workstation.

Must be connected to TAMUQ VPN to login to workstation.

By default, microphone is muted for everyone. If you have any questions, please raise your hand via Zoom and we will take the question.

During the training we will have multiple hands on exercises and anonymous surveys.

What is a Linux Container?

- Packages all the applications/binaries and dependencies into a single file called “Container Image”.
- Container image consists of custom binaries and Just Enough Operating System (JeOS).
- More efficent than full (hardware-level) virtualization.
- Offers portability and reproducibility.
- Ensures your application won’t break when you port it to a new environment.

What its good for?

- Package an analysis pipeline so that it runs on your laptop, in the cloud, and in a high-performance computing (HPC) environment to produce the same result.
- Publish a paper and include a link to a container with all of the data and software that you used so that others can easily reproduce your results.
- Install and run an application that requires a complicated stack of dependencies with a few keystrokes.
- Create a pipeline or complex workflow where each individual program is meant to run on a different operating system.

Difference between Virtual Machines and Containers

VM

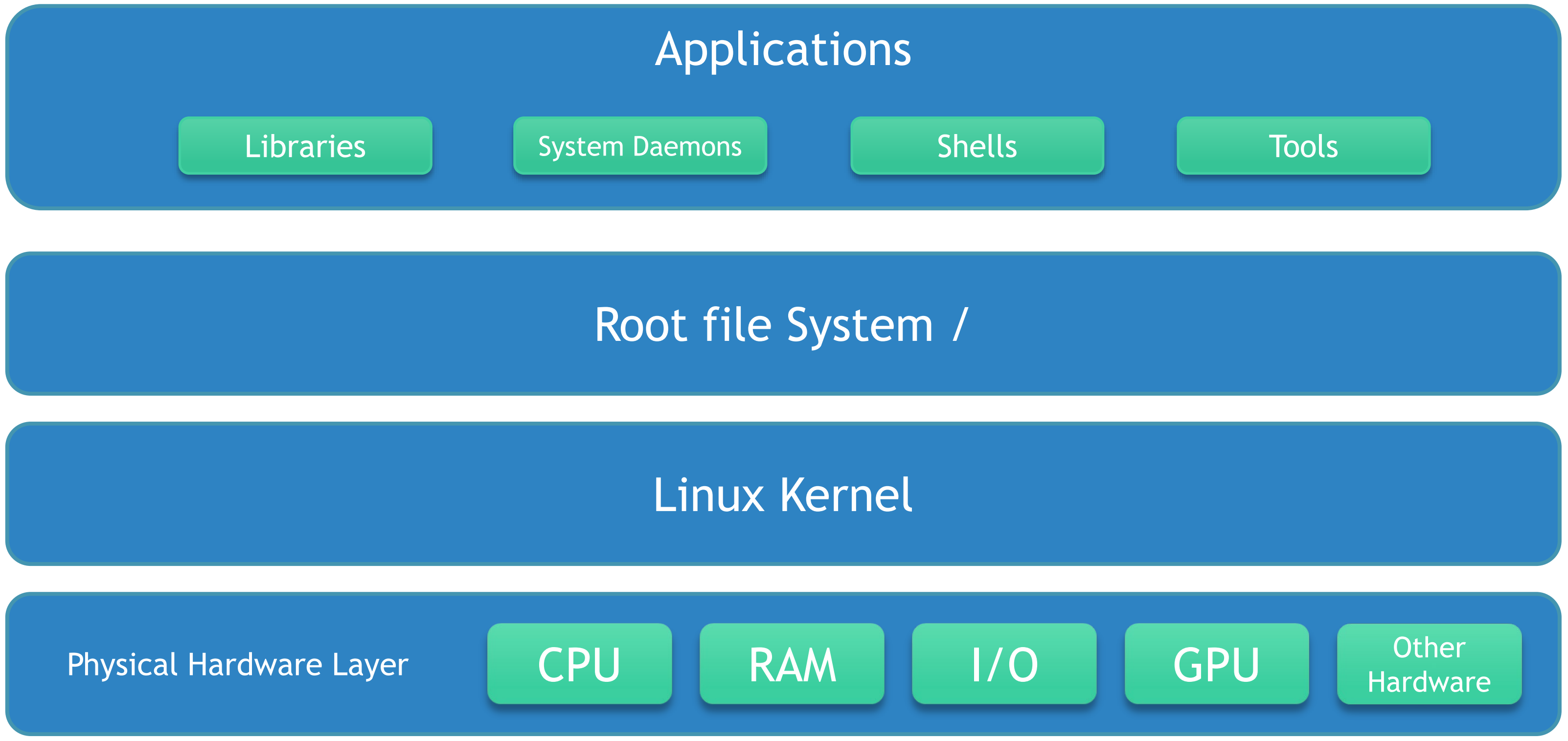
Virtual Machines

- Hardware level virtualization.
- Complete operating system right down to the kernel.
- Slow and resource hungry. Every time you launch a VM, it must bring up an entirely new OS.

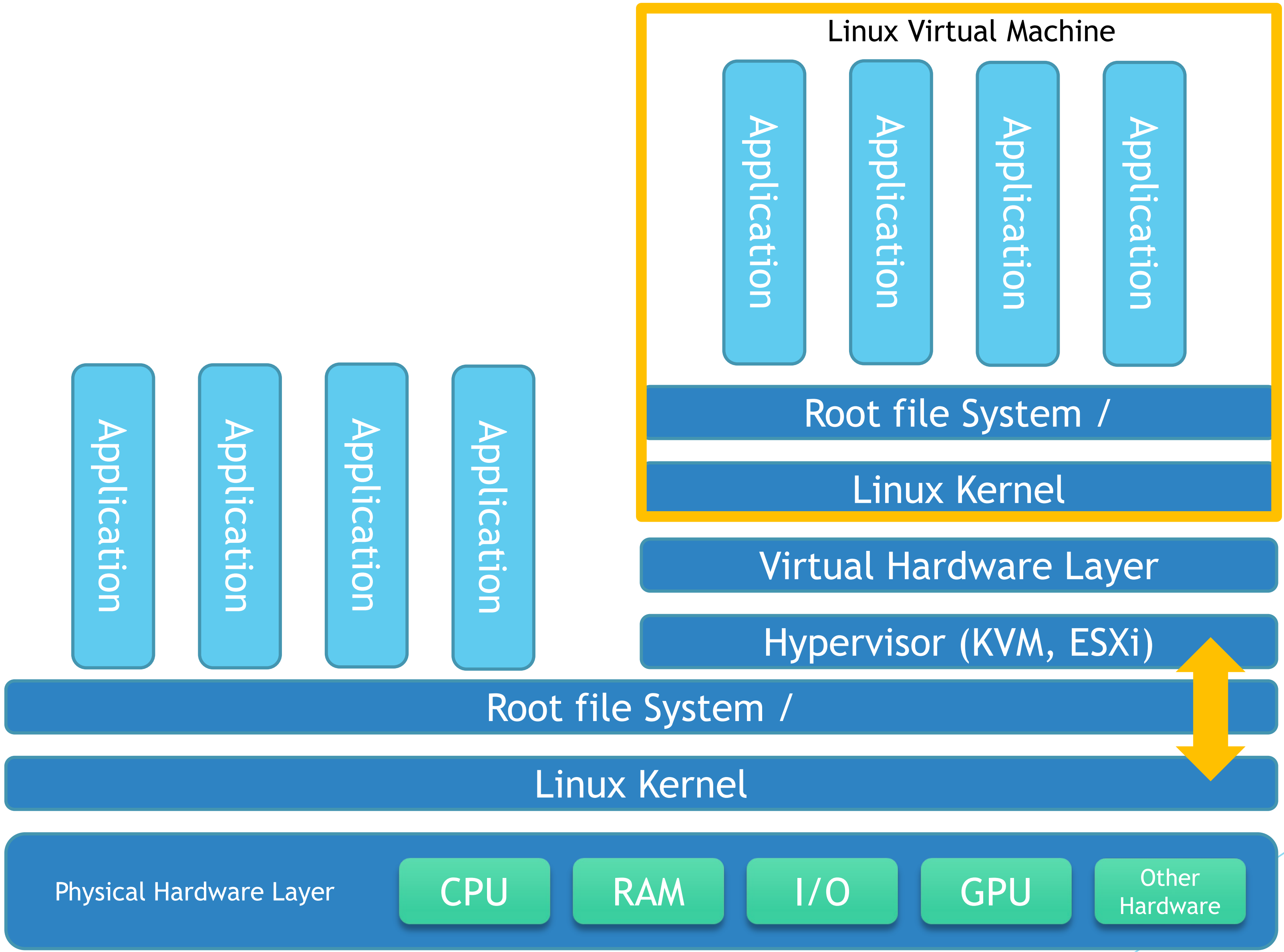
Containers

- OS Level Virtualization
- Much faster and lighter than VMs.
- Start and stop quickly and are suitable for running single apps.

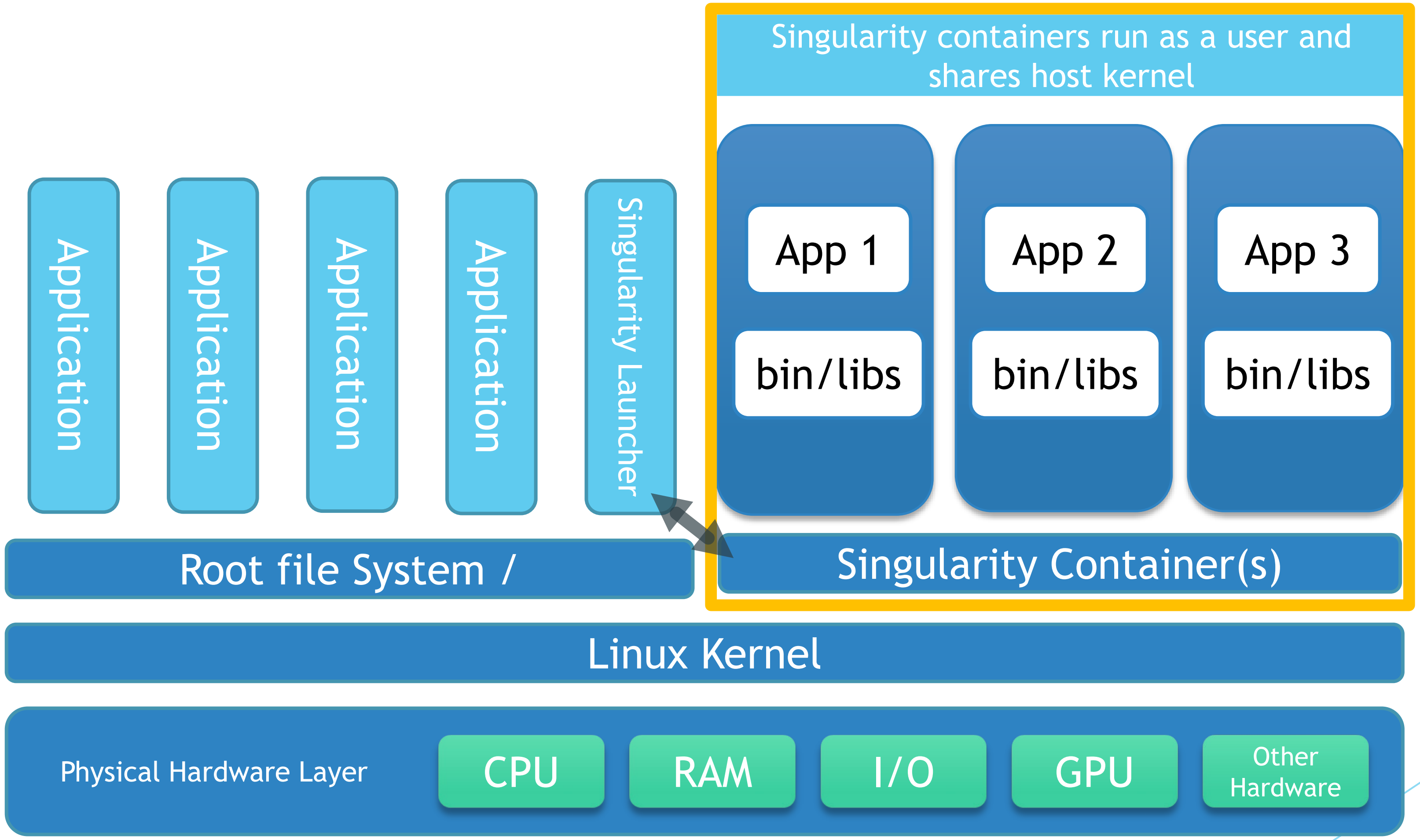
Linux OS Architecture



Linux OS with Virtual Machines



Linux OS with Singularity Containers



Common container frameworks

Docker	• Most commonly used container framework. • Not built for HPC, users can escalate sudo privileges inside the container. • Shines for DevOPs teams providing cloud-native micro services.
Shifter	• Linux containers for HPC, developed at NERSC. • Uses Docker functionality but makes it safe in shared HPC systems. • Image gateway used to convert Docker images before use.
Singularity	• Not based on Docker but can directly import/run Docker container images. • HPC oriented and designed to run in shared environments. • A user inside a Singularity container is the same user as outside the container.

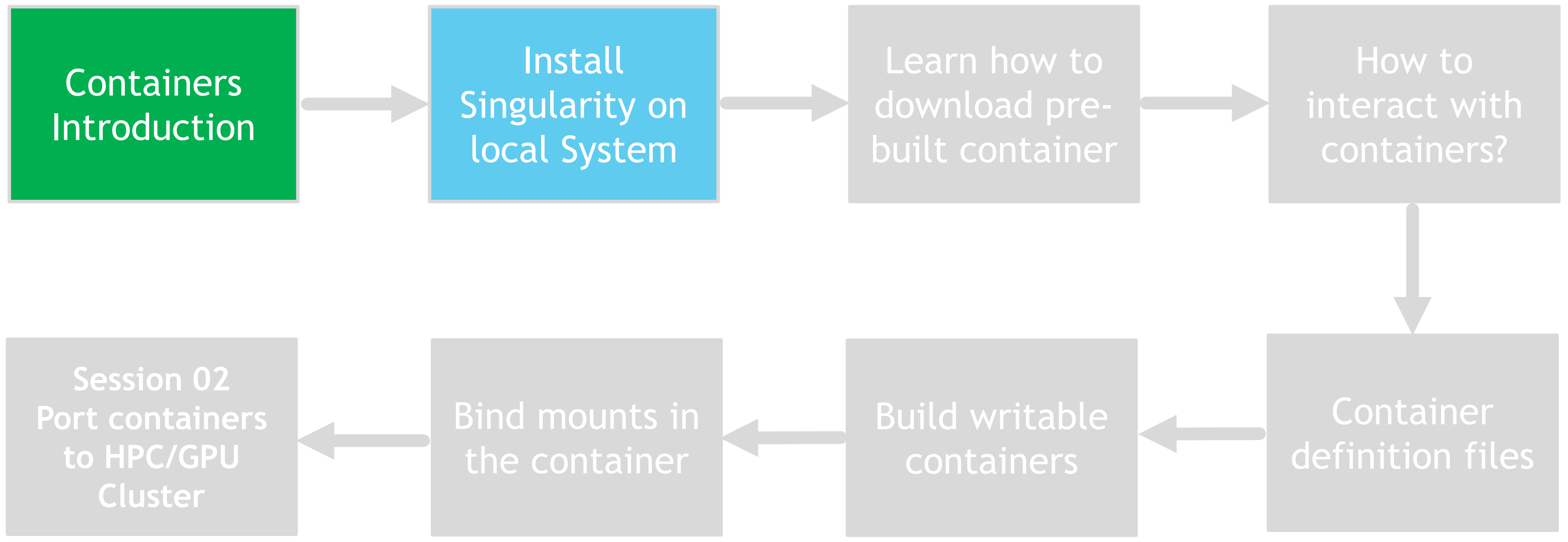
Introduction to Singularity
https://sylabs.io/

- Rleatively new container framework invented by Greg Kurtzer at Lawrence Berkley National labs.
- Developed with focus on security, scientific software and HPC systems in mind.
- Singularity assumes that each application will have its own container.
- You require a build system where you have root permissions to build a container.
- The built container is then ported to a production system where you may or may not be a root user. E.g. HPC System.

Why Singularity?

- Doesn’t requires any application daemon running on a system.
- Singularity container can be distributed as a file (singularity image file).
- Singularity exectuable can be compiled from source code.
- Can work in HPC systems where shared file system is present. Users can not access/modify data which doesn’t belongs to them.
- No changes required on HPC system or in a resource scheduler.

Session 01: Install singularity on personal workstation


Containers Introduction
Install Singularity on local System
Learn how to download pre-built container
How to interact with containers?
Container definition files
Build writable containers
Bind mounts in the container
Session 02 Port containers to HPC/GPU Cluster

Install Singularity on personal workstation

- **Your workstation runs Linux distro**
  - Compile Singularity from source code.
  - **Guide:** https://rc-docs.qatar.tamu.edu/index.php/Linux_containers#Linux
- **Your workstation runs Windows**
  - Use Sylabs provided Virtual machine in combination with Vagrant and Virtual box
  - **Guide:** https://rc-docs.qatar.tamu.edu/index.php/Linux_containers#Windows
- **Your workstation runs Mac OS**
  - Use Sylabs provided Virtual machine in combination with Vagrant and Virtual box
  - **Guide:** https://rc-docs.qatar.tamu.edu/index.php/Linux_containers#MAC_OS

Lets connect to remote workstation

In your local MobaXterm or Mac terminal, login to remote workstation using supplied credentials

```
ssh student@ip_addr

student@ws01:~$
```

Verify Singularity version available in your workstation

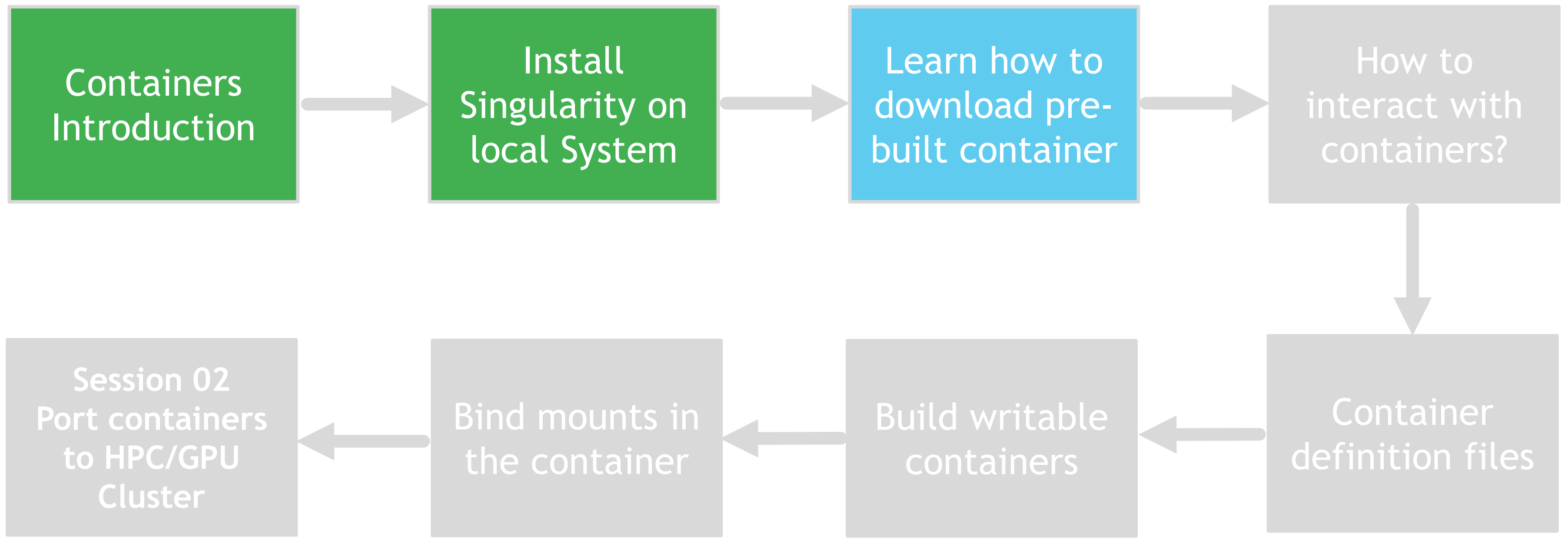
```
student@ws01:~# singularity version
```

singularity --help

Explore singularity help

```
student@ws01:~# singularity --help
singularity help <command> [<subcommand>]
Examples:
student@ws01:~# singularity help shell
```

Session 01: Download pre-built containers


Containers Introduction
Install Singularity on local System
Learn how to download pre-built container
How to interact with containers?
Container definition files
Build writable containers
Bind mounts in the container
Session 02 Port containers to HPC/GPU Cluster

Where to find pre-built containers?

Pre-built containers can be found on various online repositories

Singularity Container Library	Developed and maintained by Sylabs https://cloud.sylabs.io/library	We will explore in todays session
Docker Hub	Developed and maintained by Docker https://hub.docker.com/	We will explore in todays session
Quay.io	Developed and maintained by Red Hat https://quay.io/	
Nvidia GPU Cloud	Developed and maintained by NVIDIA https://ngc.nvidia.com/	We will explore in Session 02
BioContainers	Develped and maintained by the Bioconda group https://biocontainers.pro/	

Singularity ‘pull’ command to download pre-built containers

Use singularity “pull” command to download image from remote repository

$$\underbrace{\texttt{singularity}}_{\text{Singularity Executable}}\ \underbrace{\texttt{pull}}_{\text{Subcommand}}\ \underbrace{\texttt{image.sif}}_{\text{Desired Name of Image}}\ \underbrace{\texttt{source}}_{\text{Remote Repo Name}}\texttt{://}\underbrace{\texttt{image}}_{\text{Name of remote Image}}\texttt{:}\underbrace{\texttt{tag}}_{\text{Version required}}$$

Explore Help on Singularity Pull Command

```
student@ws01:~# singularity help pull
Examples:
```

From Sylabs cloud library

```
$ singularity pull alpine.sif library://alpine:latest
```

From Docker

```
$ singularity pull tensorflow.sif docker://tensorflow/tensorflow:latest
```

Explore remote Repos

Docker Hub

- https://hub.docker.com

Singularity Container Library

- https://cloud.sylabs.io

Hands-On Exercise 01

Task 01

Download Fedora 28 from Docker Hub and store image locally as "Fedora_28.sif"

Task 02

Download Ubuntu 18.04 from Singularity Library and store image locally as "Ubuntu_1804.sif"

Hands-On Exercise 01: Solution

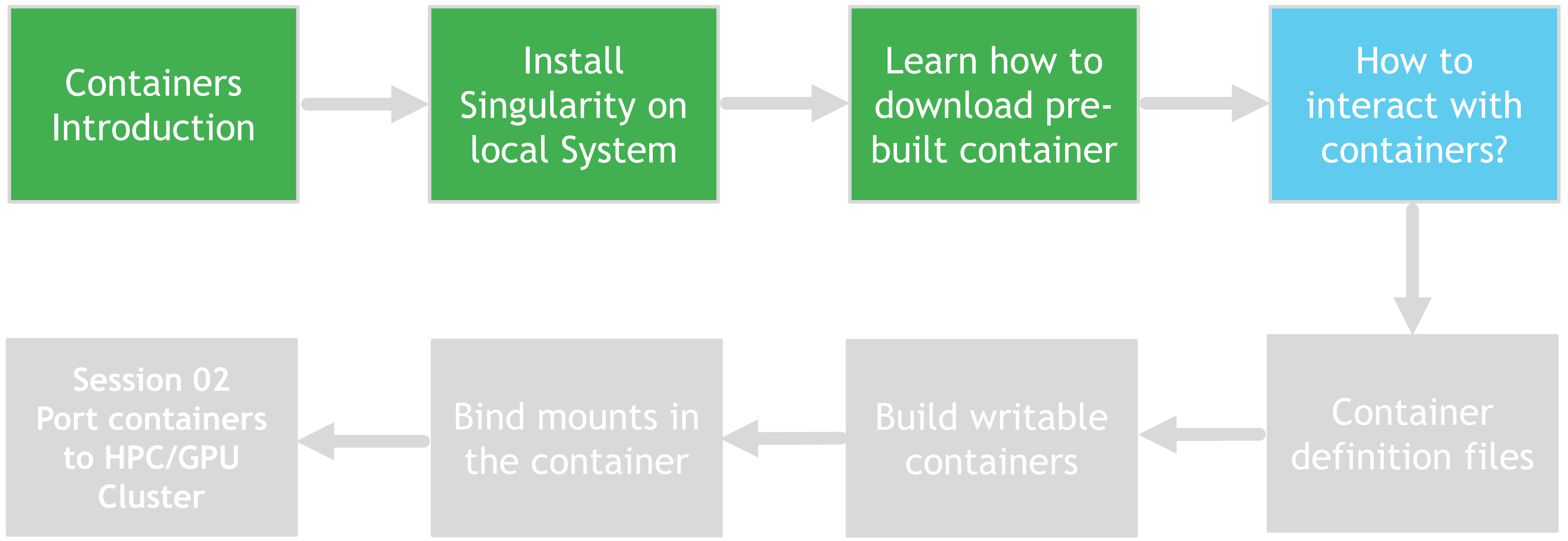
Task 01

```
singularity pull ubuntu_1804.sif library://ubuntu:18.04
```

Task 02

```
singularity pull fedora_28.sif docker://fedora:28
```

Session 01: How to interact with containers?


Containers Introduction
Install Singularity on local System
Learn how to download pre-built container
How to interact with containers?
Container definition files
Build writable containers
Bind mounts in the container
Session 02 Port containers to HPC/GPU Cluster

Interact with Containers: Shell

Use singularity “shell” command to start interactive shell inside the container

$$\underbrace{\texttt{singularity}}_{\text{Singularity Executable}}\ \underbrace{\texttt{shell}}_{\text{Sub command}}\ \underbrace{\texttt{image.sif}}_{\text{Target Image to Enter}}$$

Explore Help on Singularity Shell Command

```
student@ws01:~# singularity help shell
Examples:
  $ singularity shell /tmp/Debian.sif
  Singularity/Debian.sif> pwd
  /home/gmk/test
  Singularity/Debian.sif> exit
```

Interact with Containers: Exec

Use singularity “exec” subcommand to execute any command inside the container

$$\underbrace{\texttt{singularity}}_{\text{Singularity Executable}}\ \underbrace{\texttt{exec}}_{\text{Sub command}}\ \underbrace{\texttt{image.sif}}_{\text{Target Image}}\ \underbrace{\texttt{command}}_{\text{Command to issue}}$$

Explore Help on Singularity Exec Command

```
student@ws01:~# singularity help exec
Examples:
  $ singularity exec /tmp/debian.sif cat /etc/debian_version
  $ singularity exec /tmp/debian.sif python ./hello_world.py
```

Hands-On Exercise 02

Task 01

Start interactive session inside container Fedora_28.sif which was already downloaded. Issue following command. Note the output of the command for the upcoming assessment.

```
cat /etc/fedora-release
```

Task 02

Use singularity exec to execute below command in ubuntu_1804.sif container which was downloaded earlier.

```
cat /etc/lsb-release
```

Task 03

Use singularity exec to execute below command in ubuntu_1804.sif container which was downloaded earlier.

```
python3 --version
```

Hands-On Exercise 02: Solution

Task 01

```
student@ws01:~# singularity shell fedora_28.sif
Singularity> cat /etc/fedora-release
Fedora release 28 (Twenty Eight)
Singularity> exit
```

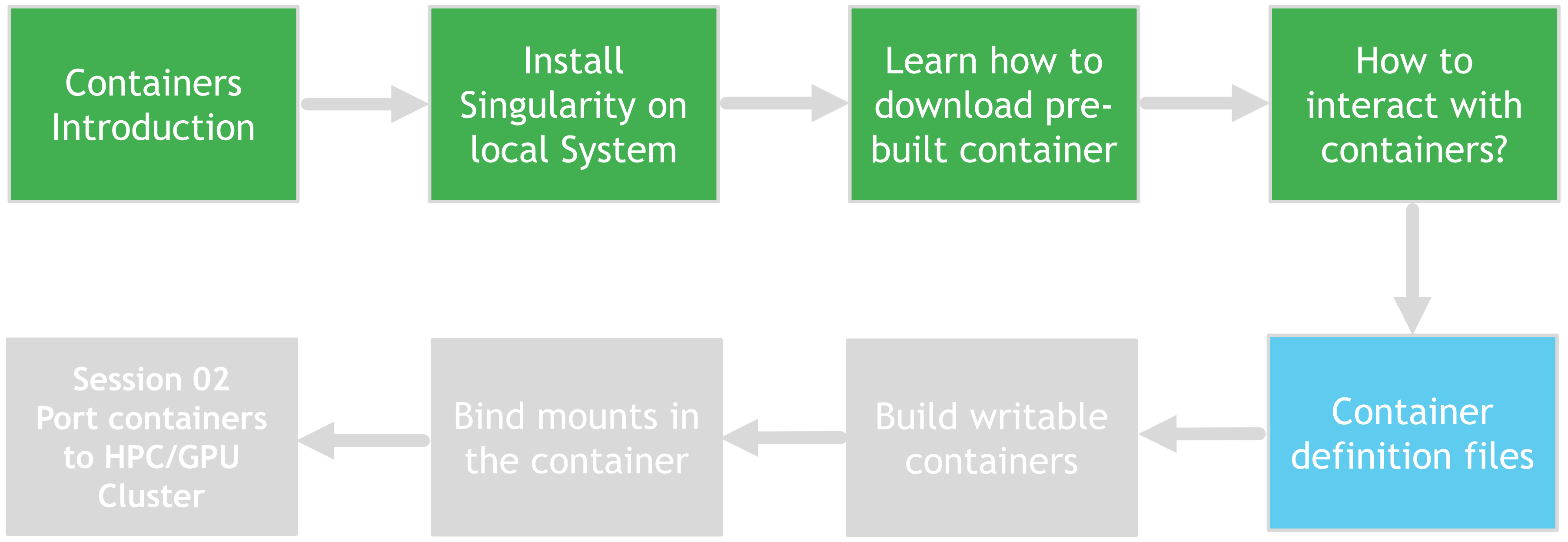
Task 02

```
student@ws01:~# singularity exec ubuntu_1804.sif cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04 LTS"
```

Task 03

```
student@ws01:~# singularity exec ubuntu_1804.sif python --version
/.singularity.d/actions/exec: 21: exec: python3: not found
```

Session 01: Container definition files


Containers Introduction
Install Singularity on local System
Learn how to download pre-built container
How to interact with containers?
Container definition files
Build writable containers
Bind mounts in the container
Session 02 Port containers to HPC/GPU Cluster

Building containers using Singularity definition files

Singularity definition files

- Set of blueprints explaining how to build a custom container.
- Includes specifics of which base container to use and what should be installed inside the container at the build time.
- Divided into two parts:
  - Header
    - Header describes the core operating system to build within the container.
  - Sections
    - Each section is defined by a % character followed by name of a section.
    - Each section performs specific tasks. Eg. File copy, Setting environment variables, Post build tasks etc.

More information: https://sylabs.io/guides/3.5/user-guide/definition_files.html

Lets have a look at a sample definition file

Sample definition file for Singularity container build (myapp.def)

```
Bootstrap: library
From: ubuntu:18.04

%environment
     export MYVAR="This is a test Container"

%post
    apt-get install -y vim

%files
    /file1 /opt

%labels
    Author user@email.com
    Version v0.0.1

%help
    This is my test application.
```

Build a container with definition file

Use singularity "build" command to build container from definition file

$$\underbrace{\texttt{singularity}}_{\text{Singularity Executable}}\ \underbrace{\texttt{build}}_{\text{Sub command}}\ \underbrace{\texttt{myapp.sif}}_{\text{Image name}}\ \underbrace{\texttt{myapp.def}}_{\text{Definition file}}$$

Explore Help on Singularity Build Command

```
student@ws01:~# singularity help build
```

Hands-On Exercise 03

Task 01

- In last exercise, when we issued "python3 --version" in ubuntu_1804.sif container, we couldn't find python3 installation inside the container.
- Now since we know how to write a definition file to modify container contents at build time. Lets write a definition file called "myapp.def" which will install python3 inside the container.
- Build container using this definition file.
- Use either singularity shell or exec to verify that the new container has python3 installed.

Remember: Building container requires sudo privileges. 'student' account has sudo privileges.

Hands-On Exercise 03: Solution

myapp.def

```
Bootstrap: library
From: ubuntu:18.04

%post
    apt-get install -y python3
```

Build container from definition file

```
student@ws01:~$ sudo singularity build myapp.sif myapp.def
```

Verify if python3 is installed inside the container

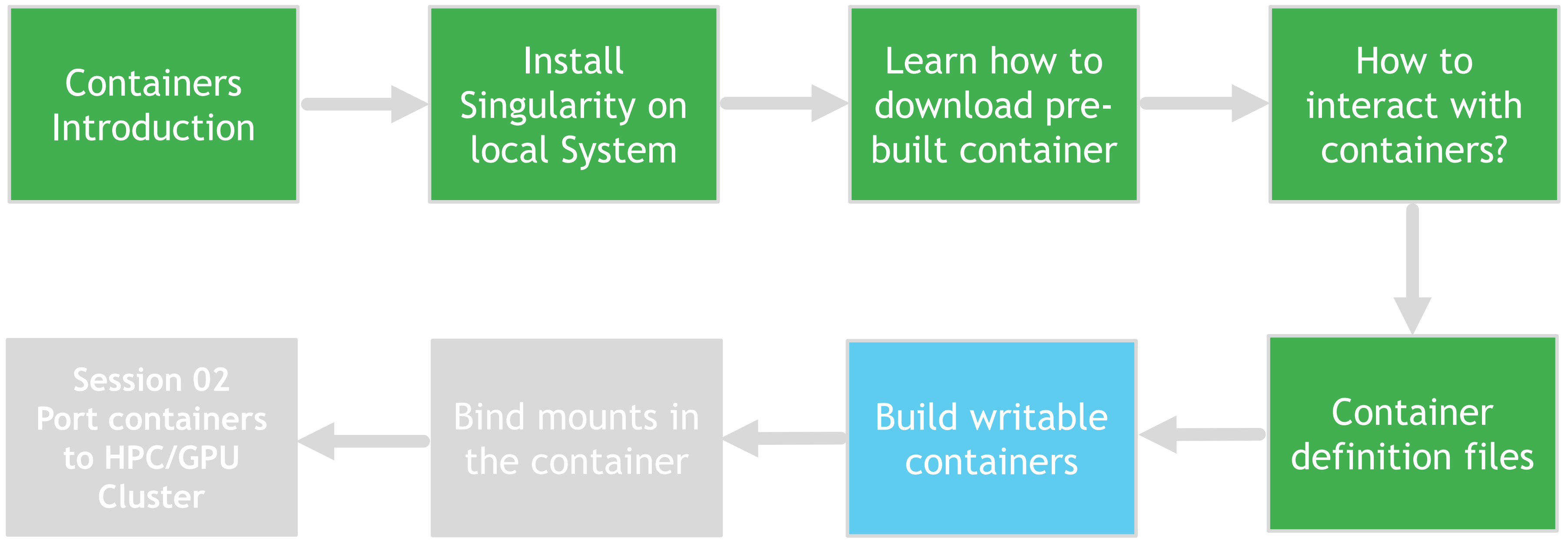
```
student@ws01:~$ singularity exec myapp.sif python3 --version

Python 3.6.5
```

Container development cycle

Steps in container development cycle

Sandbox	• Create a writable container (called a sandbox)
Connect	• Start interactive session via SHELL in the container with --writable option
Modify	• Install any required packages/applications/libraries
Record	• Record the changes in a separate file called as “definition file”
Verify	• Rebuild the container from the definition file and verify if the resultant container works as expected.
Production	• Rebuild the container as prodcution ready read-only container (.sif) image.

Session 01: What we are going to learn today?


Containers Introduction
Install Singularity on local System
Learn how to download pre-built container
How to interact with containers?
Container definition files
Build writable containers
Bind mounts in the container
Session 02 Port containers to HPC/GPU Cluster

Writable containers

- A container can be modified if it was build as a “sandbox”.
- A writable container is called as a “sandbox” container.
- Sandbox containers are meant for development and testing only. They should never be used in a production environment.

Building a sandbox container

Use singularity "build" command with --sandbox flag and use definition file

$$\underbrace{\texttt{singularity}}_{\text{Singularity Executable}}\ \underbrace{\texttt{build}}_{\text{Sub command}}\ \underbrace{\texttt{--sandbox}}_{\text{Sub command}}\ \underbrace{\texttt{myapp}}_{\text{Dir name}}\ \underbrace{\texttt{myapp.def}}_{\text{Definition file}}$$

Use singularity "build" command with --sandbox flag and use remote repo

$$\underbrace{\texttt{singularity}}_{\text{Singularity Executable}}\ \underbrace{\texttt{build}}_{\text{Sub command}}\ \underbrace{\texttt{--sandbox}}_{\text{Sub command}}\ \underbrace{\texttt{myapp}}_{\text{Dir name}}\ \underbrace{\texttt{source}}_{\text{Remote Repo Name}}\texttt{://}\underbrace{\texttt{image}}_{\text{Name of remote Image}}\texttt{:}\underbrace{\texttt{tag}}_{\text{Version required}}$$

Interacting with sandbox container

- When you build a container with --sandbox flag, the resultant container is a directory instead of singularity image file (.sif).
- You can interact with this sandbox container just like you did with singularity image file.
  - `singularity shell myapp`
  - `singularity exec myapp cat /etc/lsb-release`
- By default when you do a shell or exec inside a sandbox container, you are not allowed to make any changes.
- You will need to add --writable flag to make changes inside the sandbox container.

Making changes to sandbox container

Use singularity shell with --writable flag

$$\underbrace{\texttt{singularity}}_{\text{Singularity Executable}}\ \underbrace{\texttt{shell}}_{\text{Sub command}}\ \underbrace{\texttt{--writable}}_{\text{Sub command}}\ \underbrace{\texttt{sandboxdir}}_{\text{Sandbox Container}}$$

Use singularity exec with --writable flag

$$\underbrace{\texttt{singularity}}_{\text{Singularity Executable}}\ \underbrace{\texttt{exec}}_{\text{Sub command}}\ \underbrace{\texttt{--writable}}_{\text{Sub command}}\ \underbrace{\texttt{sandboxdir}}_{\text{Sandbox Container}}\ \underbrace{\texttt{command}}_{\text{Command}}$$

Convert sandbox container to .sif

Use singularity build to convert sandbox container to singularity image file

$$\underbrace{\texttt{singularity}}_{\text{Singularity Executable}}\ \underbrace{\texttt{build}}_{\text{Sub command}}\ \underbrace{\texttt{myapp.sif}}_{\text{Output Image}}\ \underbrace{\texttt{myapp}}_{\text{Sandbox container}}$$

Hands-On Exercise 04

Task

- Your manager has given you a source code which has to be run on HPC system. But the HPC admin has informed you that the required dependencies can not be met on HPC system due to various technical reasons. Since you attended Introduction to Linux container course, you can easily solve this problem by installing all the requirements in a container and then port container to HPC system.
- HPC Admin told you that he can help you port the container on HPC system, but he wants you to share container build definition file, so he is aware of what is happening inside the container.
- App requirements are below;
  - **Base OS:** Centos 8
  - **Libraries:** fftw-libs, fftw-devel, openmpi.x86_64, openmpi-devel.x86_64, boost
  - **Packages:** Python3, vim
- Deliverables
  - Definition file which can reproduce exact same container.
- *What will be your approach to build a container?*

Hands-On Exercise 04: Approach

Step 01

- Build a sandbox container based on Centos 8.

Step 02

- Do an interactive SHELL to sandbox container and install requirements.
- While you are installing requirements, make sure to take a note of all the actions you are performing inside the container.

Step 03

- Translate those requirements into a definition file

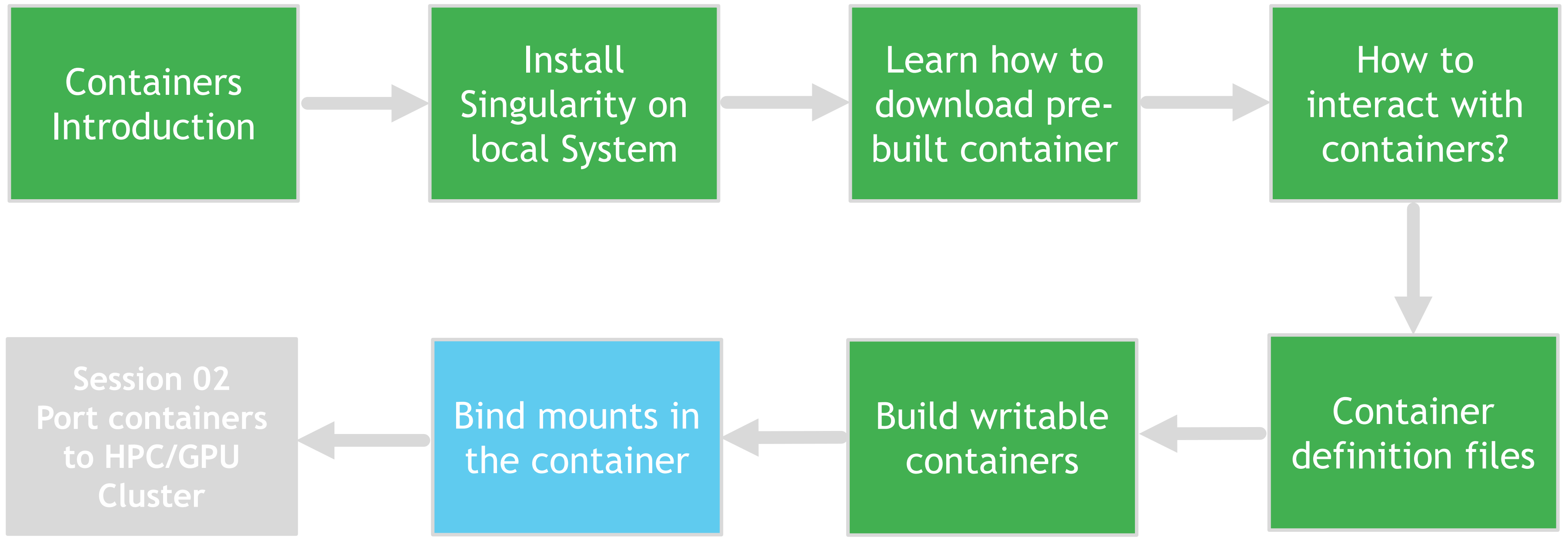
Step 04

- Rebuild container with definition file to make sure that definition file is working as expected.

Step 05

- Convert the sandbox container to singularity image file (.sif) and port it to the HPC System.

Session 01: What we are going to learn today?


Containers Introduction
Install Singularity on local System
Learn how to download pre-built container
How to interact with containers?
Container definition files
Build writable containers
Bind mounts in the container
Session 02 Port containers to HPC/GPU Cluster

How to modify/access files of host system from within the container?

- When you launch a container, your home directory and few other directories are mounted inside the container by default.
  - $HOME, /tmp, /proc, /sys, /dev
- It is also possible to mount other host directories inside the container using --bind option or the environmental variable $SINGULARITY_BINDPATH
- You can bind directories as read only or read write with bind options.

More information: https://sylabs.io/guides/3.5/user-guide/bind_paths_and_mounts.html

Examples

- On Host system, create a directory

```
$ mkdir /data
$ echo "This file is on host system" >> /data/file.info
```

- Bind host directory inside a container at /mnt path

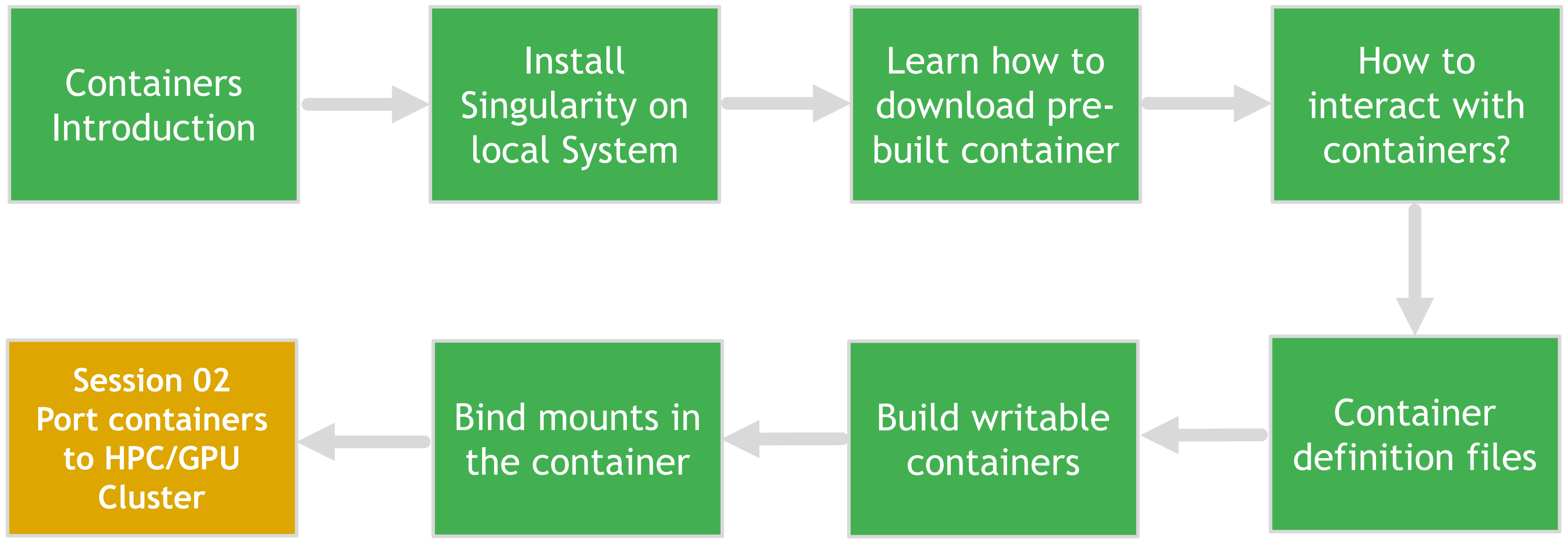
```
$ singularity exec --bind /data:/mnt myapp.sif cat /mnt/file.info
This file is on host system
```

- You can also bind multiple directories in single command with this syntax:

```
$ singularity shell --bind /data1:/mnt1,/data2/mnt2 myapp.sif
```

More information: https://sylabs.io/guides/3.5/user-guide/bind_paths_and_mounts.html

Session 01: What we are going to learn today?


Containers Introduction
Install Singularity on local System
Learn how to download pre-built container
How to interact with containers?
Container definition files
Build writable containers
Bind mounts in the container
Session 02 Port containers to HPC/GPU Cluster

Session 02: Porting production ready containers to HPC or GPU Cluster

- Remote building a container
- Porting container to raad2
- Container with SLURM in batch job
- Packaging MPI application in container
- Containers with Nvidia GPUs
- Running containers from Nvidia GPU cloud