# INTRODUCTION TO LINUX CONTAINERS FOR HPC

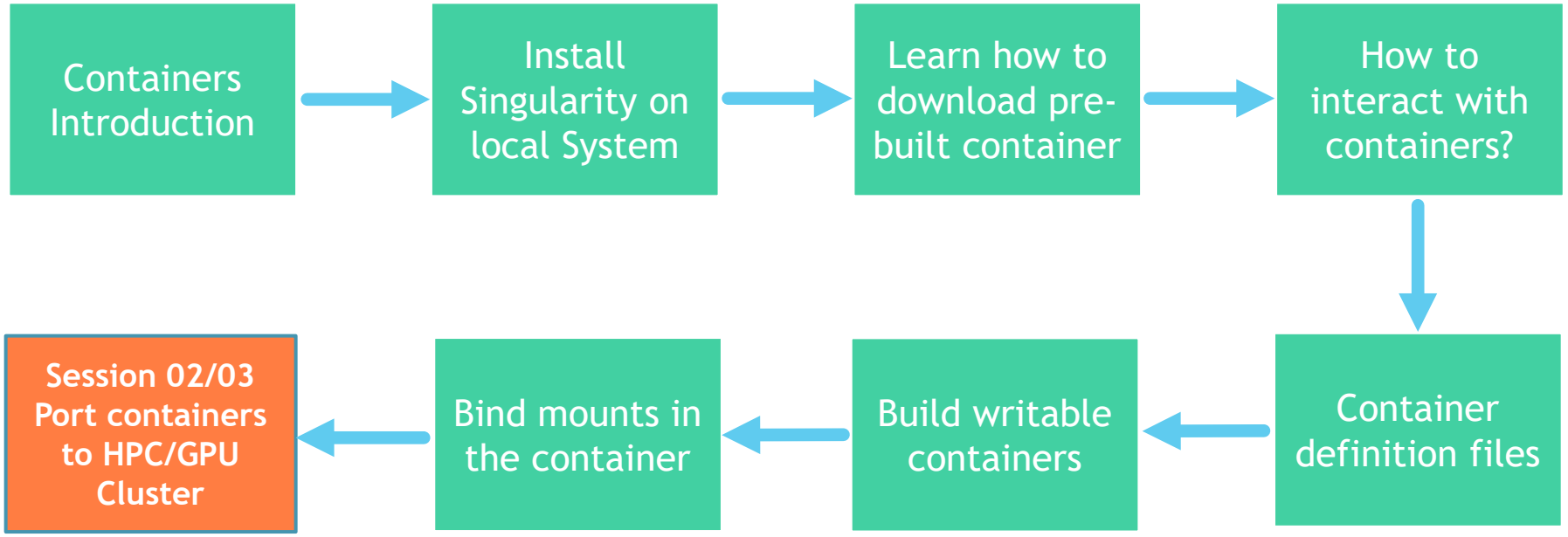
## *SESSION 02*

Mustafa Arif

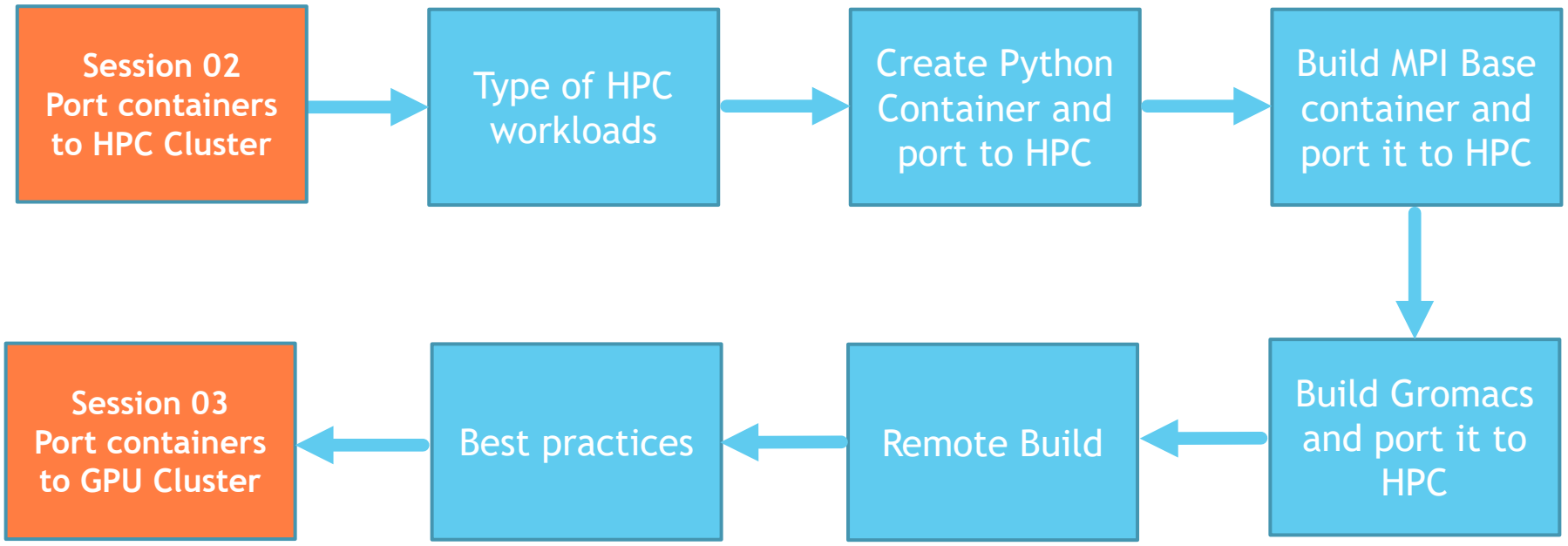
mustafa.arif@qatar.tamu.edu

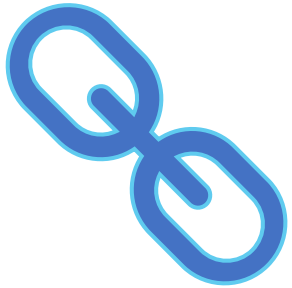
Texas A&M University at Qatar

# Session 01: What we have learned?



# Session 02: What we are going to learn today?





# Documentation Link

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

# Training guidelines

Dev instances for Singularity are provided in cloud.

Attendees should have access to HPC system “raad2” for Session 02.

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.

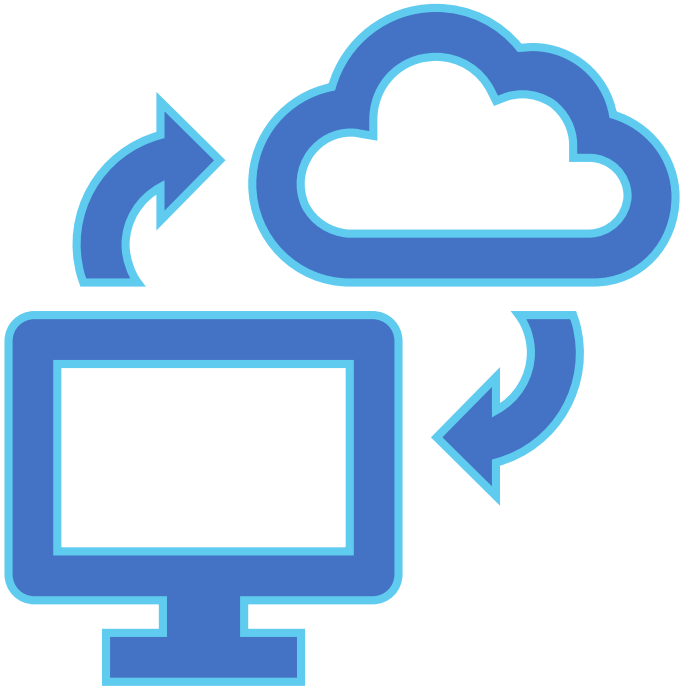
# Lets connect to remote workstation

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

```
ssh -p <port> student@x.cloudapp.azure.com
student@host:~$
```

Verify Singularity version available in your workstation

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



# Connect to raad2

# Connect to raad2

In your local MobaXterm or Mac terminal, login to raad2

```
ssh user@raad2.qatar.tamu.edu
student@host:~$
```

Clone git repo for this tutorial

```
user@raad2a:~# git clone https://github.com/mustafaarif/workshops.git
```

# Session 01: Recap

## Introduction to Linux Containers

- Offers portability, reproducibility.
- More efficient than hardware-level virtualization.
- Light weight and shares kernel with host operating system.
- Common container frameworks are Docker, Shifter and Singularity.
- Shifter is more suited for HPC environments.

## Installing Singularity on local workstation

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

# Session 01: Recap

- How to download pre-built containers?
  - From Docker
    - `singularity pull ubuntu_1804.sif docker://ubuntu:18.04`
  - From Singularity Library
    - `singularity pull ubuntu_1804.sif library://ubuntu:18.04`
- How to interact with the containers?
  - SHELL
    - `singularity shell ubuntu_1804.sif`
  - Execute Command inside container
    - `singularity exec ubuntu_1804.sif cat /etc/lsb-release`

# Session 01: Recap

```
Bootstrap: library
From: ubuntu:18.04

%environment
  export PY_VER="3"

%post
  apt-get install python3

%labels
  Author mustafa.arif@qatar.tamu.edu
  Version v0.0.1

%help
  This container is built on top of Ubuntu 18.04 and have
python3 installed
```

- Singularity definition files
  - Blueprint of a container.
  - Can reproduce exact same software stack.
  - `singularity build image.sif blueprint.def`
- Writable containers
  - `singularity build --sandbox ubuntu docker://ubuntu:18.04`
- Bind-Mount in the containers
  - Allows you to mount host directories inside the container.
    - `singularity shell --bind /dir_out:/dir_in myapp.sif cat /dir_in/myfile`

# HPC Workloads

## Serial workloads

- Runs on a single core and is mutually exclusive of other tasks.
- A single node on raad2 can host up to 24 serial jobs of same or multiple users.

## Multi-threaded workloads

- Requires more than a single core, but computation is still bound to single node.

## MPI Applications

- Problem is divided into small chunks and distributed over various nodes.
- Communication across nodes happens via MPI or specific methods.
- Most of the scientific codes have MPI Support.

# Generic workflow for porting applications to raad2

- Gather requirements for your application
  - Is it a serial or multi-node application?
  - Does vendor/author suggest any specific base operating system? Ubuntu/Centos/Fedora etc.
- On development workstation (with sudo access), create sandbox container with base OS.
  - Use trusted sources to download the base container. E.g. Docker/Sylabs
- Install prerequisites inside sandbox container.
- Download/Copy source of the target application inside the container and perform the build.
- Test your application before converting into .sif image.
- Copy the final .sif image to raad2.
- Verify application working with SLURM scheduler in interactive or batch mode.

# MPI Support with Singularity

- Message Passing Interface (MPI) is extensively used in HPC to communicate across various nodes.
- There are two main open-source implementations of MPI;
  - MPICH
  - OpenMPI
- Singularity supports both implementations and offers two different ways to use MPI with HPC system.
  - Hybrid Model
    - MPI is installed inside the container.
  - Bind Model
    - MPI installation on host system is mounted inside the container at run time.
- More information
  - https://sylabs.io/guides/3.5/user-guide/mpi.html

# Case Study 01: Porting Python container to raad2

- **Requirements**
  - **Base OS:** Ubuntu
  - **Python Version:** 3.8.2
  - **Python packages:** Tensorflow
  - **OS packages:** wget, git, vim, nano
- **Deliverables**
  - Singularity Image file
  - Definition file

# Case Study 02: Build a base MPI container with MPICH

- **Requirements**
  - **Base OS:** Ubuntu
  - **MPICH Ver:** 3.3
- **Deliverables**
  - Singularity Image file
  - Definition file

# Case Study 03: Build Gromacs with MPI Support and port it to raad2

- GROMACS is a versatile package to perform molecular dynamics, i.e. simulate the Newtonian equations of motion for systems with hundreds to millions of particles.
- This package is actively used by many in raad2 community.
- Package updates are released often, and each new release offers better performance features.
- It is difficult for admins to provide latest package on HPC system regularly.
- Latest releases often depends on various other updated packages. Compiling all the dependencies is very time consuming and it can take few days to weeks.

# Case Study 03: Build Gromacs with MPI Support and port it to raad2

- Build Approach 01: Identify if there is any repo which provides latest builds of Gromacs containers. You can download that container hoping that it is optimized to your needs.
- Build Approach 02: If you want to build from scratch;
  - Install required pre-reqs. A good start is to install at least build-essential, cmake and gfortran on ubuntu.
  - Install mpich (raad2 supports mpich, so its recommended to use mpich)
  - Build Gromacs.
  - Copy container to raad2 via scp
- Make container act as a native application on raad2.
- Launch as batch job with SLURM.

# Remote building a container

- Allows users to perform a remote build of a container without requiring sudo privileges.
- The remote build happens in cloud at cloud.sylabs.com
- You will need to create an account on https://cloud.sylabs.com
- Remote builder accepts definition file only.
- `singularity build --remote myapp.sif myapp.def`

# Let's remote build a container

# Best Practices

# Navigating documentation

https://sylabs.io/guides/3.5/user-guide/

# Questions?