Containers for HPC Reproducibility: Building, Deploying, and Optimizing

Carlos del-Castillo-Negrete

Research Associate, Texas Advanced Computing Center

Slide Deck Available at:

https://github.com/cdelcastillo21/tamu_hpc_containers

Workshop Outline

- 9:00 AM - 9:45 AM: Introduction to Containers in HPC
  - Overview of container technology
  - Benefits of using containers in HPC environments
  - Comparison of Docker and Singularity/Apptainer.
- 9:45 AM – 10:00 AM: Break (finish set-up)
- 9:30 AM – 10:30 AM: Building & Running Your First HPC Container
  - Building a simple container using Docker
  - Running container locally
  - Running and managing containers on HPC systems
- 10:30 AM – 10:45 AM: Break
- 10:45 AM – 11:15 AM: GPU Containers
  - Overview of GPU support in containers
  - Using NVIDIA GPUs with containers
- 11:15 AM - 11:30 AM: Break
- 11:30 AM - 12:00 AM: Q&A and Wrap-Up OR mini-Hackathon
  - Open floor for questions and discussion OR
  - Mini-Hackathon – Build and run a container of your choosing

What you'll need

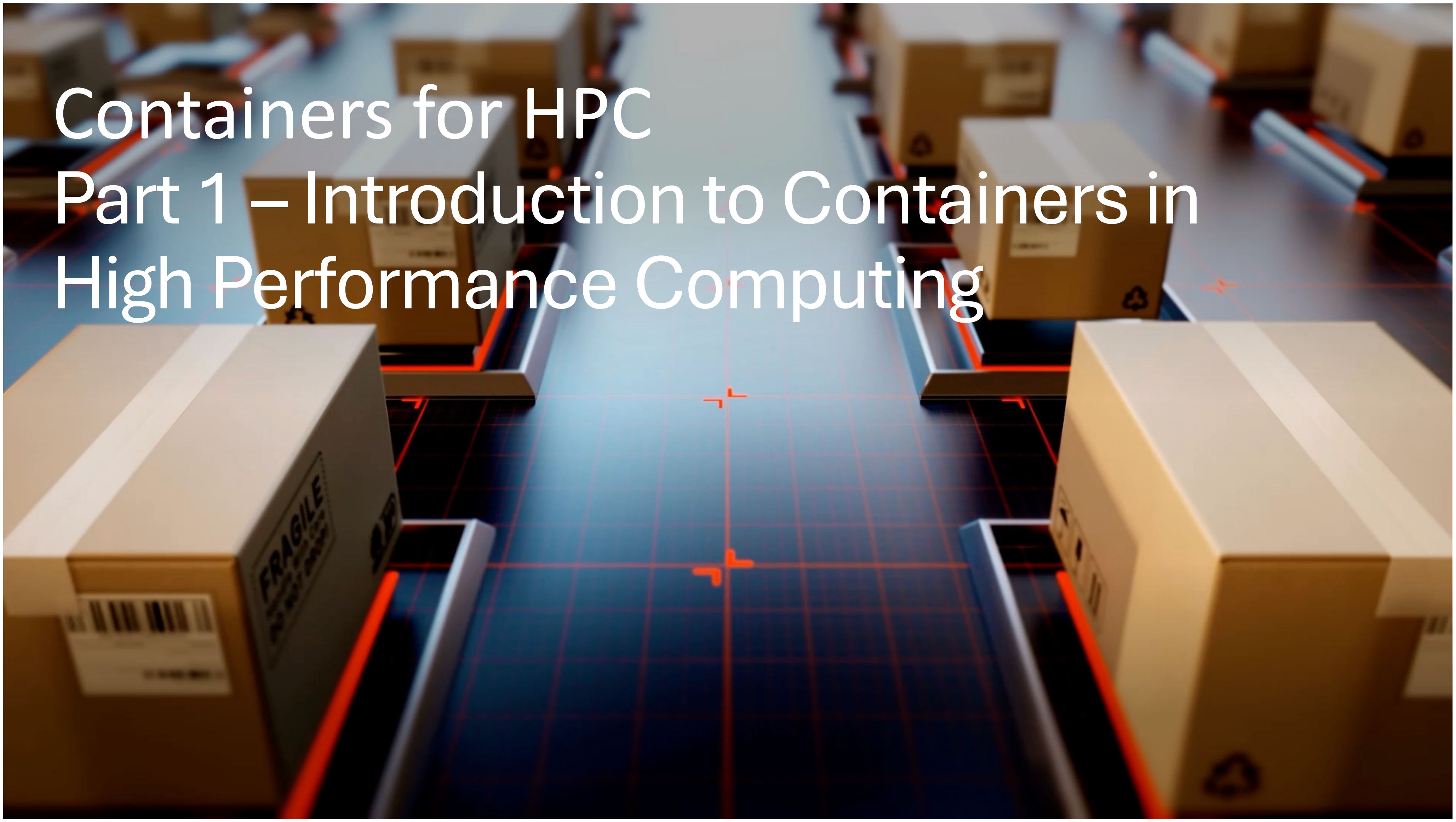
- Docker
  - **Docker Engine (required)** - https://docs.docker.com/engine/install/
  - Preferably Docker Desktop (optional) :
    - Mac - https://docs.docker.com/desktop/install/mac-install/
    - Windows - https://docs.docker.com/desktop/install/windows-install/
  - Docker hub account (required) - https://hub.docker.com
  - Docker Build cloud (alternate to building locally) - https://build.docker.com
- **Local editor and Terminal Environment**
  - VSCode - Recommended best cross-platform solution – https://code.visualstudio.com
  - MobaXTerm– Alternative Terminal for windows users - https://mobaxterm.mobatek.net
- **Access to FASTER via web portal (required)** :
  - https://portal-faster-access.hprc.tamu.edu/
  - https://portal-faster.hprc.tamu.edu

Learning Resources and Links

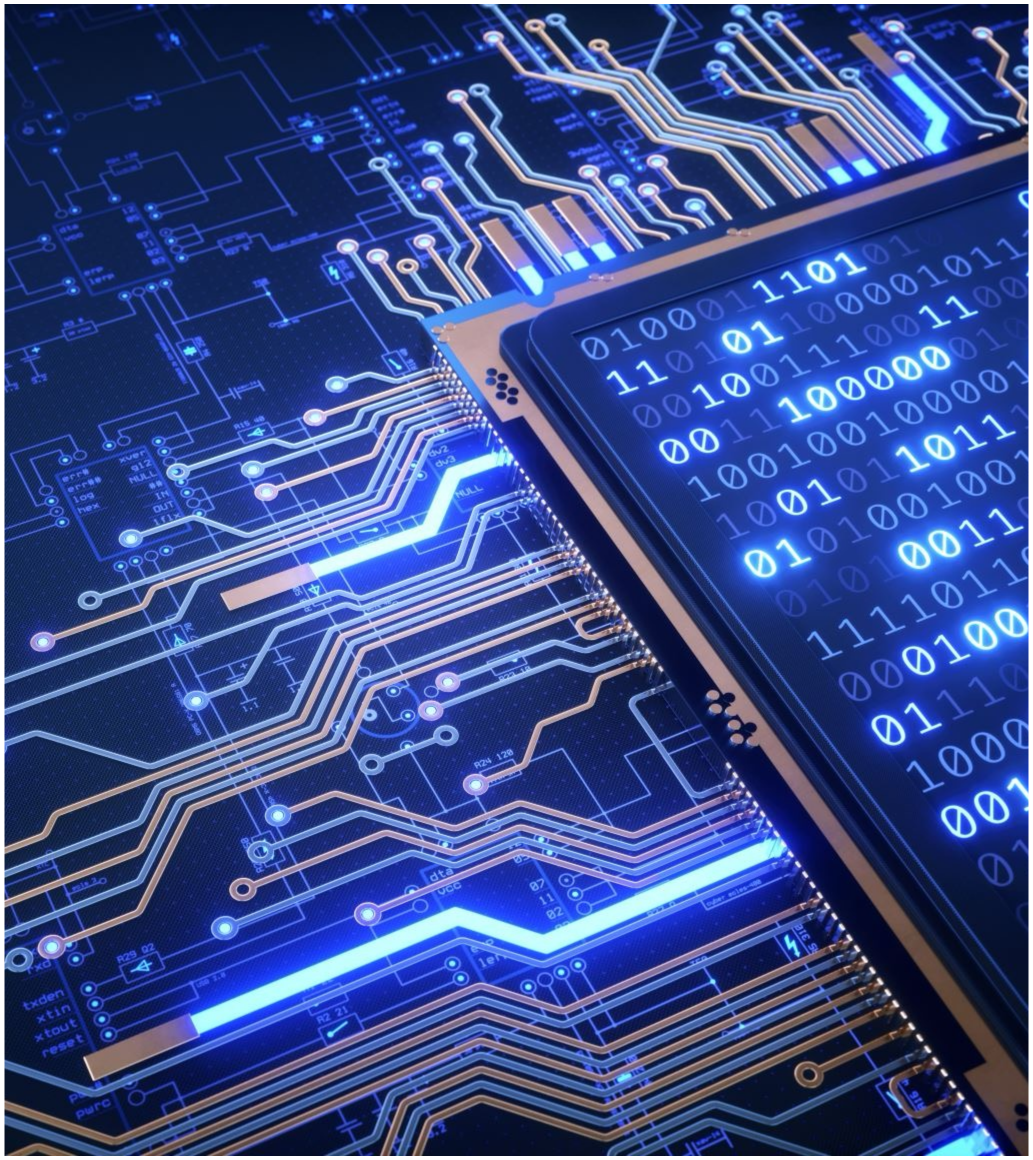
- TAMU HPRC:
  - HPRC Wiki https://hprc.tamu.edu/wiki/SW:Singularity
  - HPRC on Youtube https://www.youtube.com/c/TexasAMHPRC
  - HPRC ACES Container tutorials - https://hprc.tamu.edu/training/aces_containers_scientific.html
- Singularity Manual https://apptainer.org/user-docs/3.8/
- Docker Manual https://docs.docker.com/
- Other container courses:
  - NBIS - https://nbis-reproducible-research.readthedocs.io/en/latest/singularity/
  - Arizona https://learning.cyverse.org/projects/Container-camp-2020/
  - TACC https://learn.tacc.utexas.edu/mod/page/view.php?id=95

Remember – Ask Questions!

Ok - Ready?



Containers for HPC
Part 1 – Introduction to Containers in High Performance Computing

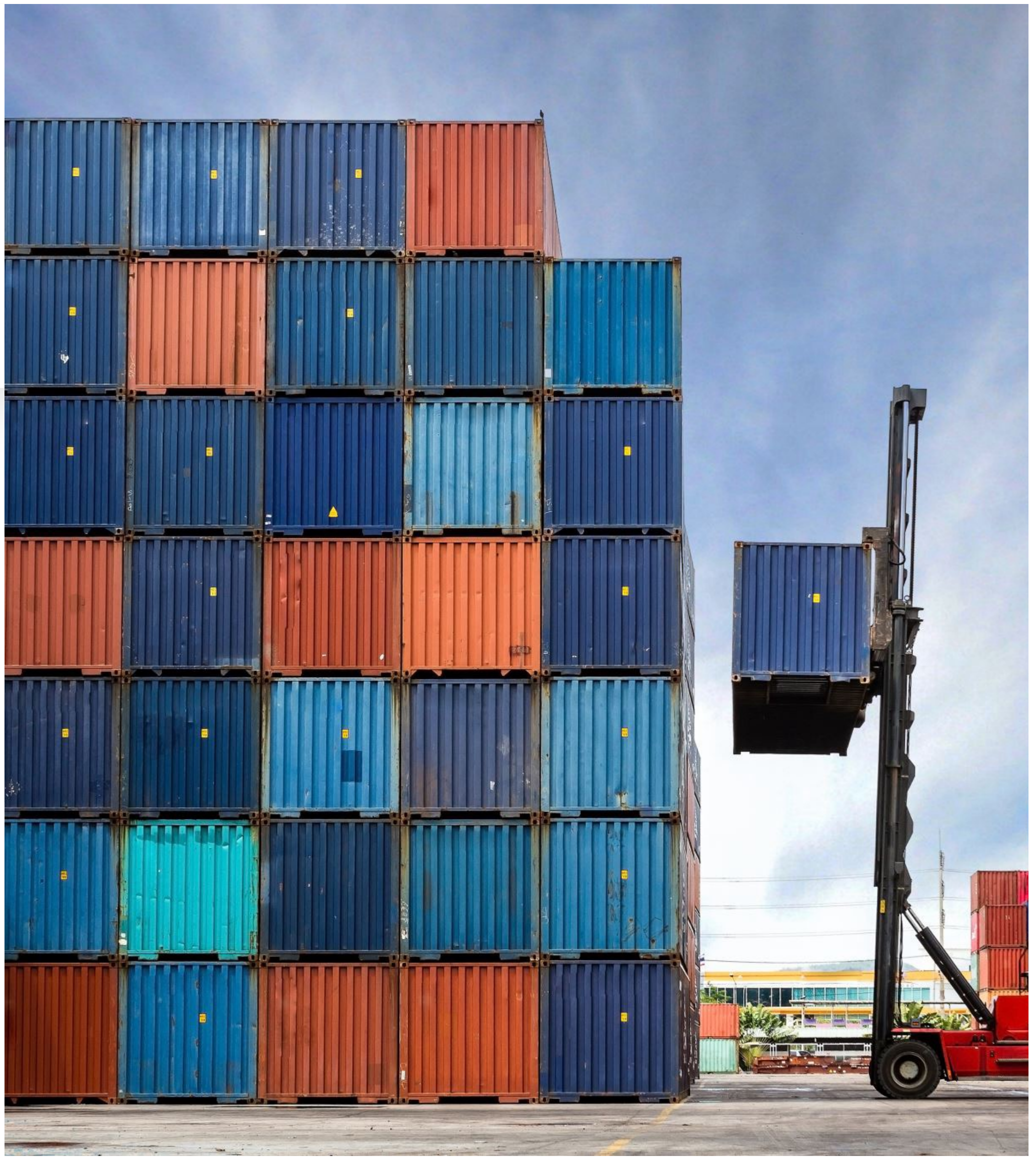


Introduction to Containers in HPC: Outline

- Overview of container technology
- Benefits of using containers in HPC environments
- Comparison of Docker, Singularity/Apptainer
- Security considerations and best practices

Containers – What's the fuss?

- Containers: A Vital Tool for Modern Computing
  - Application Development
  - Web Services
  - Scientific Computing
- Packaging Made Easy
  - Bundle applications with all dependencies
- Consistency Across Environments
  - Ensure* isolation, consistency, and reproducibility
- Deployment Without Borders:
  - A leap towards platform-agnostic deployment

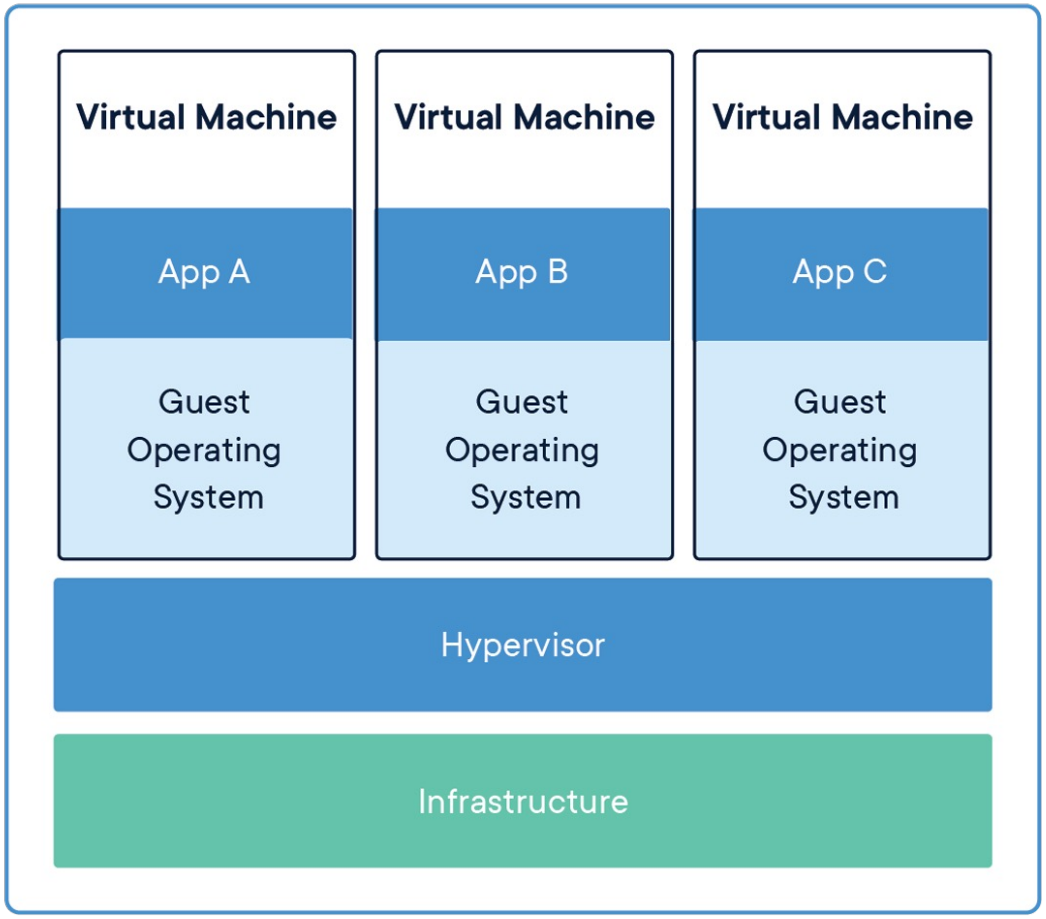


HPC Research Computing Use Cases

Enhanced reproducibility	• Difficult to build code? Do it once, containerize, and then use built container to run in multiple places.
Local development	• Test small before using HPC resources
Interactive visualization environments	• JupyterLab with custom python kernels for data analysis and visualizations

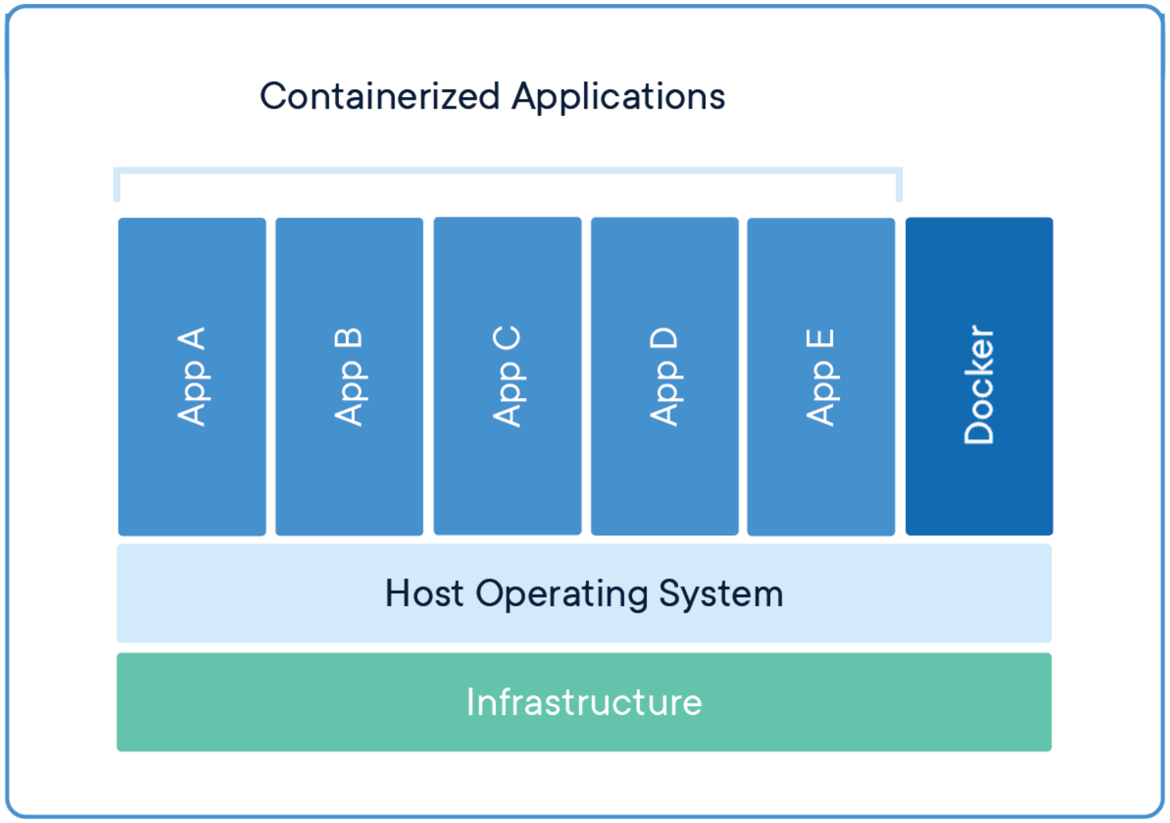
But wait, what about Virtual Machines (VMs)?

- VMs were the traditional solution to the portability issue. Only problem
  - High overhead when running on top of a hypervisor.
  - VMs take up more disk space and have long start up times.
  - Makes managing multiple VMs on the same infrastructure difficult.



Enter Containers

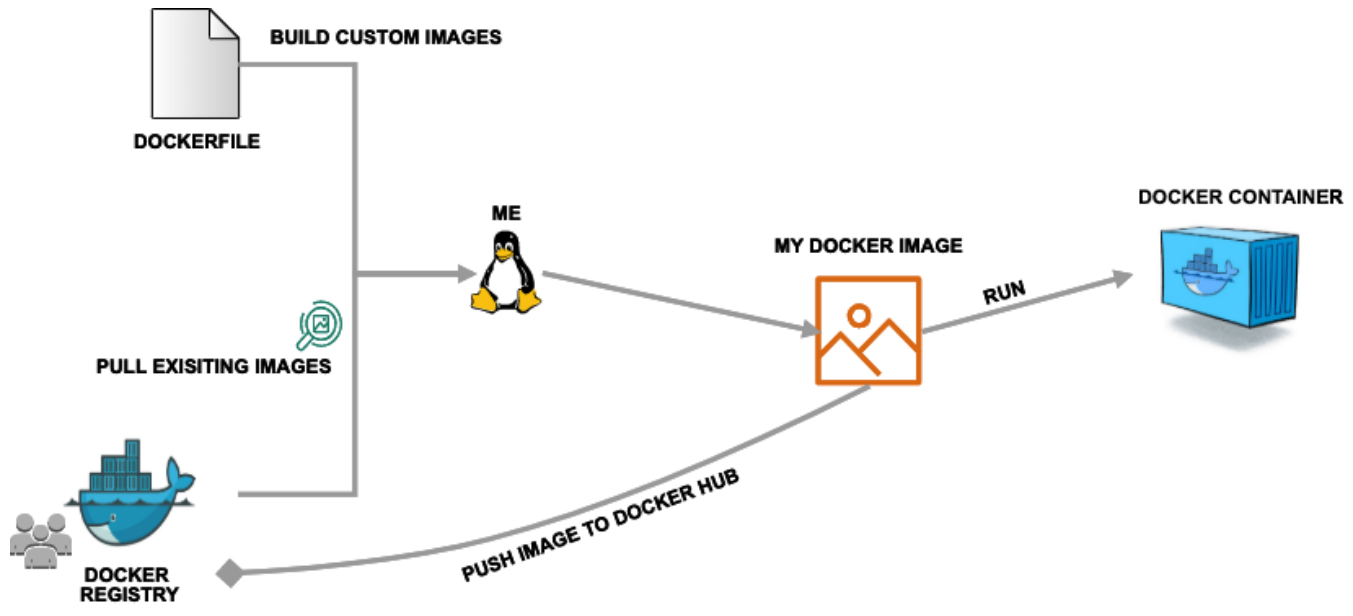
- Containers on the other hand:
    - Run on host OS
    - Take up less disk space
- A variety of containerization tools exist:
    - In cloud solutions – Docker, Podman, CharlieCloud
    - In HPC solutions – Singularity, Apptainer



Containerization Technologies

- Docker
  - Most popular, but requires root privileges.
  - Most HPC containerization platforms are compatible with Docker images.
- Singularity (now Apptainer)
  - Designed to have bare-metal performance, and does not require root permissions for running (does for building).
  - Secure, portable, and 100% reproducible in HPC environments.
- Workflow – Use Docker to build and test locally, and use Singularity to run on HPC.

Container Workflow - Local



Docker Overview

- One of the earliest container solutions created.
- Components
  - Docker Engine - Core Service for creating and managing containers.
  - Docker Hub – Repository for sharing and storing image containers
  - Docker Desktop – UI provided in docker desktop to help manage builds, containers, volumes and more.
- Concepts
  - Dockerfile
  - Image vs Container
  - Registry and Image Tags

Core concepts - Dockerfile

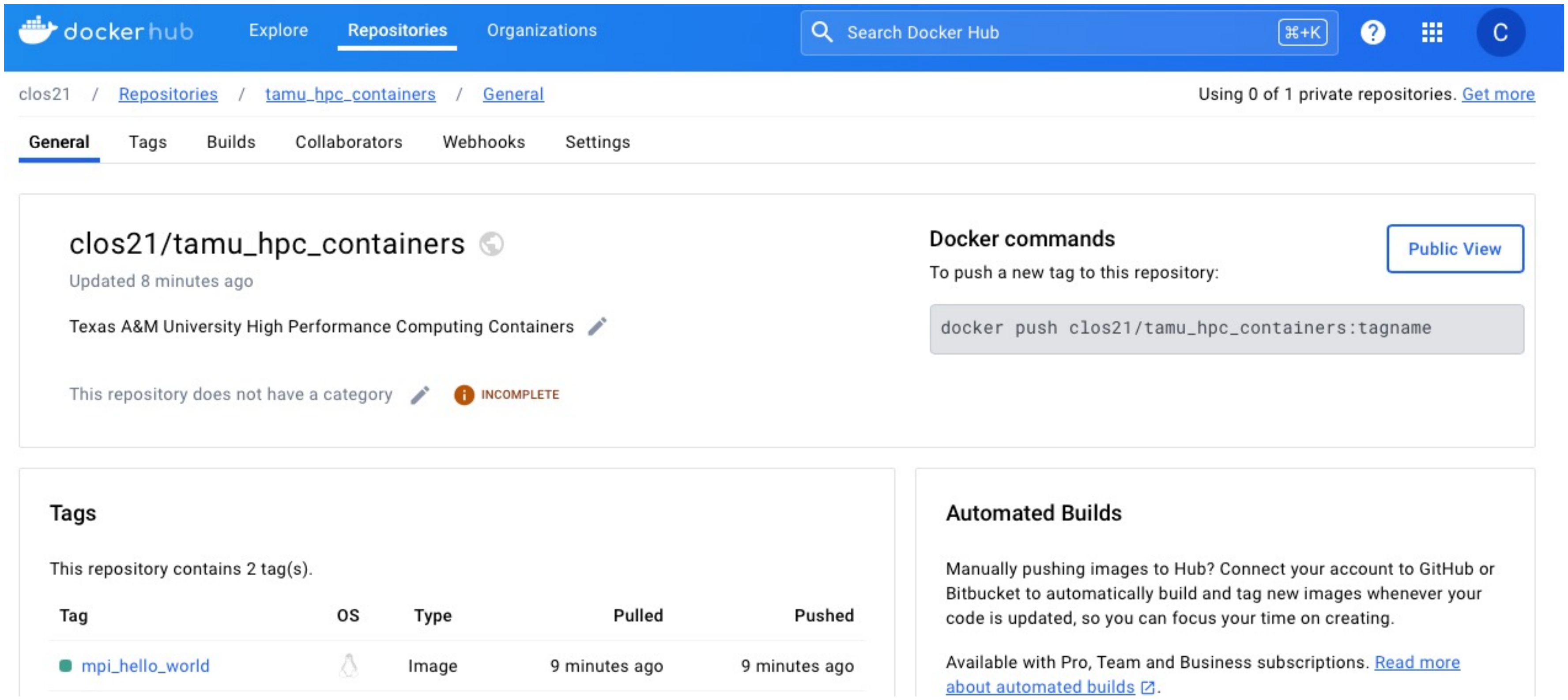
- The recipe for creating Docker Images.
- Sequence of commands that "build" a container image, with keywords like:
    - FROM – Base image to start
    - RUN – run a bash command, for example to install software.
    - COPY – Copy data from outside the container inside of it
    - ENTRYPOINT – Define what should be the entrypoint script of the container when launched.

```dockerfile
FROM ubuntu:20.04

# Install necessary packages
RUN apt-get update && apt-get install -y \
    build-essential \
    openmpi-bin \
    openmpi-common \
    libopenmpi-dev

# Copy the MPI "Hello World" script into the container
COPY mpi_hello_world.c /mpi_hello_world.c

# Compile the MPI "Hello World" script
RUN mpicc -o /mpi_hello_world /mpi_hello_world.c

# Set the entrypoint to run the MPI program
ENTRYPOINT ["mpirun", "-np", "4", "/mpi_hello_world"]
```

Core Concepts - Building Image

```
docker image build –t clos21/tamu_hpc_containers:mpi_hello_world -–push .
```

Core Concepts - Image Registry



clos21 / Repositories / tamu_hpc_containers / General

Using 0 of 1 private repositories. Get more

clos21/tamu_hpc_containers

Updated 8 minutes ago

Texas A&M University High Performance Computing Containers

This repository does not have a category

INCOMPLETE

Docker commands

To push a new tag to this repository:

```
docker push clos21/tamu_hpc_containers:tagname
```

Tags

This repository contains 2 tag(s).

Tag	OS	Type	Pulled	Pushed
mpi_hello_world		Image	9 minutes ago	9 minutes ago

Automated Builds

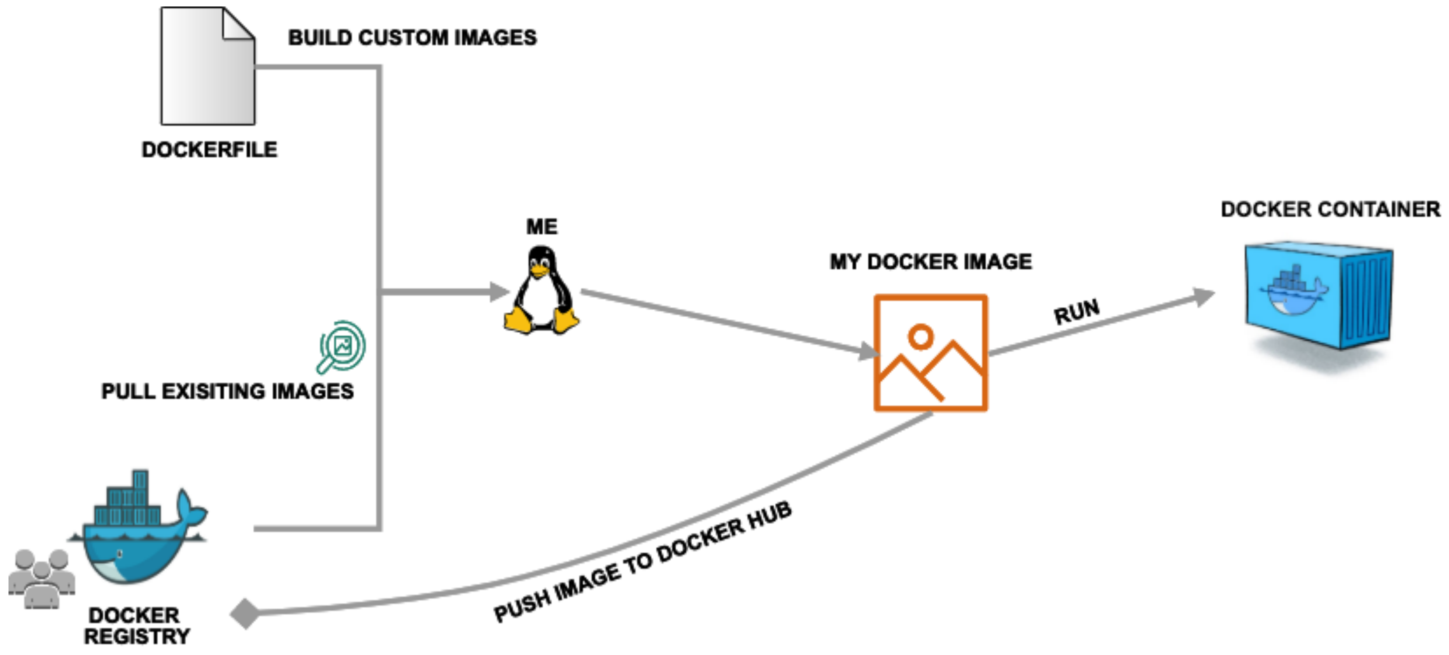
Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. Read more about automated builds.

Core Concepts – Running Container

```
docker run --rm mpi-hello-world
```

Container Workflow - Local



Running on HPC? – Enter Singularity

Singularity can fetch images from the docker hub registry and convert it to a singularity format with the ‘singularity pull’ command.

```
singularity pull docker://clos21/tamu_hpc_containers:mpi_hello_world
```

Docker vs Singularity – Host Directories

Docker: None by default. Use -v <source>:<destination> to mount a source host directory to an arbitrary destination within the container.

Singularity: Mounts your current working directory, $HOME directory, and some system directories by default. Other defaults may be set in a system-wide configuration. The --bind flag is supported to provide similar functionality as –v in docker

Docker vs Singularity – User ID

Docker: Defined in the Dockerfile, but containers run as root unless a different user is defined or specified on the command line. This user ID only exists within the container, and care must be taken when working with files on the host filesystem to make sure permissions are set correctly.

Singularity: Containers are run in “userspace”, so you are the same user and user ID both inside and outside the container.

Docker vs Singularity – Image Format

Docker: Containers are stored in layers and managed in a repository by Docker. The 'docker images' command will show you what containers are on your local machine and images are always referenced by their repository and tag name.

Apptainer: Containers are files. Apptainer can build a container on the fly if you specify a repository, but ultimately they are stored as individual files, with all the benefits and dangers inherent to files.

Containers for HPC

Part 2 – Building and Running your First Container

Building a Docker Image –MPI Hello World

Locally clone the github repo:

```
git clone https://github.com/cdelcastillo21/tamu_hpc_containers
```

Navigate to the the first example directory, and build the docker file:

```
docker image build \
        -t clos21/tamu_hpc_containers:mpi_hello_world \
        .
```

Testing container locally:

We use the docker run command to launch an image (making it a container):

```
docker run –rm clos21/tamu_hpc_containers:hello_world
```

Testing On HPC - IMPORTANT Set-up

Return to your tutorial directory (if necessary)

```
cd $SCRATCH/s_tutorial
```

Set your singularity cache directory for temporary files

```
export SINGULARITY_CACHEDIR=$TMPDIR
```

Connect to the internet for fetching images:

```
module load WebProxy
```

Docker – Cloud Build

```
docker buildx build \
        --builder cloud-clos21-tamu-hpc  \
        -t clos21/tamu_hpc_containers:mpi_hello_world \
        --load \
        --push \
        .
```

Docker Cross Platform Build (Using Cloud Builder)

```
docker buildx build \
	--platform=linux/amd64,linux/arm64 \
	--builder cloud-clos21-tamu-hpc  \
	-t clos21/tamu_hpc_containers:mpi_hello_world \
	--load \
	--push \
	.
```

Containers for HPC

Part 3 – MPI Containers

HPC Container Technology Gotchas

- HPC Systems have high-speed, low latency networks that have special drivers.
  - Infiniband, Aries, and OmniPath are three different specs for these types of networks.
  - When running MPI jobs, if the container doesn't have the right libraries, it won't be able to use those special interconnects to communicate between nodes.
- This means that MPI containers don't provide as much portability between systems.

HPC MPI – Getting things right

- By default = the network is the same inside and outside the container, the communication between containers usually just works.
- Container needs the right set of MPI libraries to interact with HPC high-speed fabrics.
- MPI is an open specification, but there are several implementations (OpenMPI, MVAPICH2, and Intel MPI to name three) with some non-overlapping feature sets.
- Different hardware implementations (e.g. Infiniband, Intel Omnipath, Cray Aries) that need to match what is inside the container. If the host and container are running different MPI implementations, or even different versions of the same implementation, MPI may not work.
- **General rule:** version of MPI inside the container to be the same version or newer than the host

Containers for HPC

Part 3 – GPU Containers

Using Singularity containers with GPU

Ensure your container is built with an adequate version of CUDA (>=11)

- Adding the --nv flag to the usual singularity command you use.
- NVIDEA container registry:
  - https://catalog.ngc.nvidia.com/
- Docker containers with GPU – search for “gpu” tag

Tensorflow GPU Exercise

- Image file: pytorch_23.09-py3.sif
- from docker://nvcr.io/nvidia/pytorch:23.09-py3

```
singularity pull docker://nvcr.io/nvidia/pytorch:23.09-py3
```

- To run:

```
singularity exec –np pytorch_23.09-py3.sif  nvidia-smi
singularity exec --nv pytorch_23.09-py3.sif python3 -c "import torch; print(torch.cuda.device_count())"
```

Appendix A

More info – VMs vs Containers

VMs vs Containers – When to use each?

- Isolation and Security - VMs win
  - VMs offer a stronger level of security, since there is a full separation of the OS kernels.
  - Containers rely on host OS security.
- Resource Efficiency – Containers win
  - Containers have less overhead
  - Are better suited for dense deployments
- Start-up Times – Containers win
  - Containers can start in milliseconds, VMs take minutes
- Portability – Containers win
  - More portable, since setting up VM hypervisor is in itself a complex process.

Appendix B

Docker – Advanced Topics

Docker Buildx vs Docker Build

- Multi-platform Builds:
    - Buildx: Supports building multi-platform images using QEMU.
    - Build: Limited to building images for the host's architecture.
- Extended Functionality:
    - Buildx: Provides advanced features like cache imports/exports and build drivers.
    - Build: Basic build functionalities without extended features.
- Enhanced Performance:
    - Buildx: Can leverage BuildKit for faster and more efficient builds.
    - Build: Traditional Docker build process, potentially slower.
- Experimental Features:
    - Buildx: Supports experimental features and new build capabilities (such as cloud builds)
    - Build: Standard and stable features without experimental options

Appendix C

HPC Containers – Advanced Topics

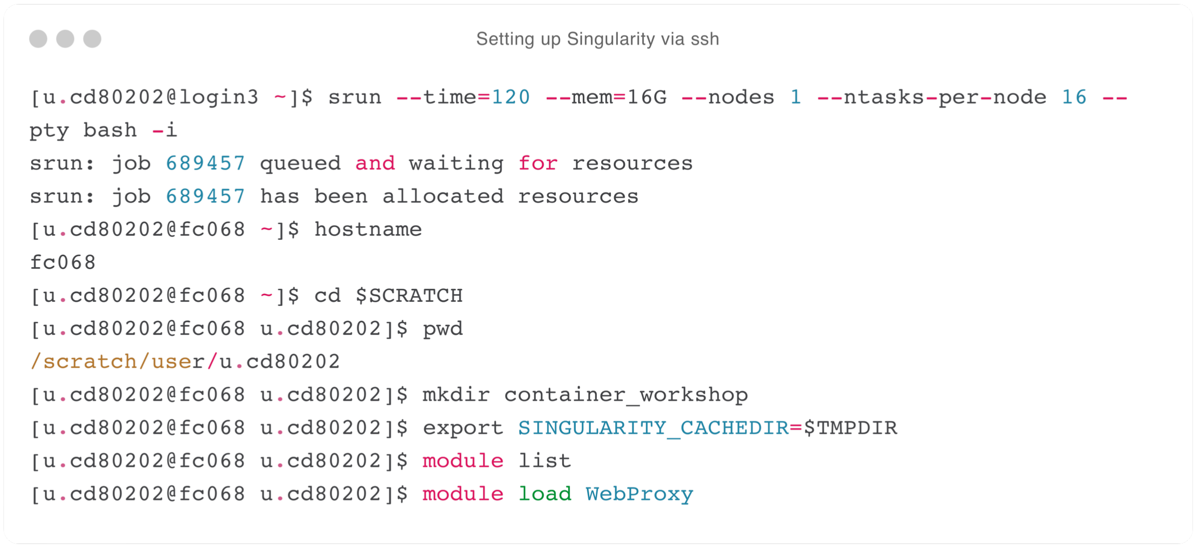
HPC Container Technology Gotchas

- HPC Systems have high-speed, low latency networks that have special drivers.
  - Infiniband, Aries, and OmniPath are three different specs for these types of networks.
  - When running MPI jobs, if the container doesn’t have the right libraries, it won’t be able to use those special interconnects to communicate between nodes.
- This means that MPI containers don’t provide as much portability between systems.

HPC MPI – Getting things right

- By default = the network is the same inside and outside the container, the communication between containers usually just works.
- Container needs the right set of MPI libraries to interact with HPC high-speed fabrics.
- MPI is an open specification, but there are several implementations (OpenMPI, MVAPICH2, and Intel MPI to name three) with some non-overlapping feature sets.
- Different hardware implementations (e.g. Infiniband, Intel Omnipath, Cray Aries) that need to match what is inside the container. If the host and container are running different MPI implementations, or even different versions of the same implementation, MPI may not work.
- **General rule:** version of MPI inside the container to be the same version or newer than the host

Launching Singularity – Command LIne



```
[u.cd80202@login3 ~]$ srun --time=120 --mem=16G --nodes 1 --ntasks-per-node 16 --
pty bash -i
srun: job 689457 queued and waiting for resources
srun: job 689457 has been allocated resources
[u.cd80202@fc068 ~]$ hostname
fc068
[u.cd80202@fc068 ~]$ cd $SCRATCH
[u.cd80202@fc068 u.cd80202]$ pwd
/scratch/user/u.cd80202
[u.cd80202@fc068 u.cd80202]$ mkdir container_workshop
[u.cd80202@fc068 u.cd80202]$ export SINGULARITY_CACHEDIR=$TMPDIR
[u.cd80202@fc068 u.cd80202]$ module list
[u.cd80202@fc068 u.cd80202]$ module load WebProxy
```