HPC Technology

(containers)

3rd Latin American Introductory School on Parallel Programming and Parallel Architecture for HPC

Dr. Fernando Posada

Assoc. Research Professor

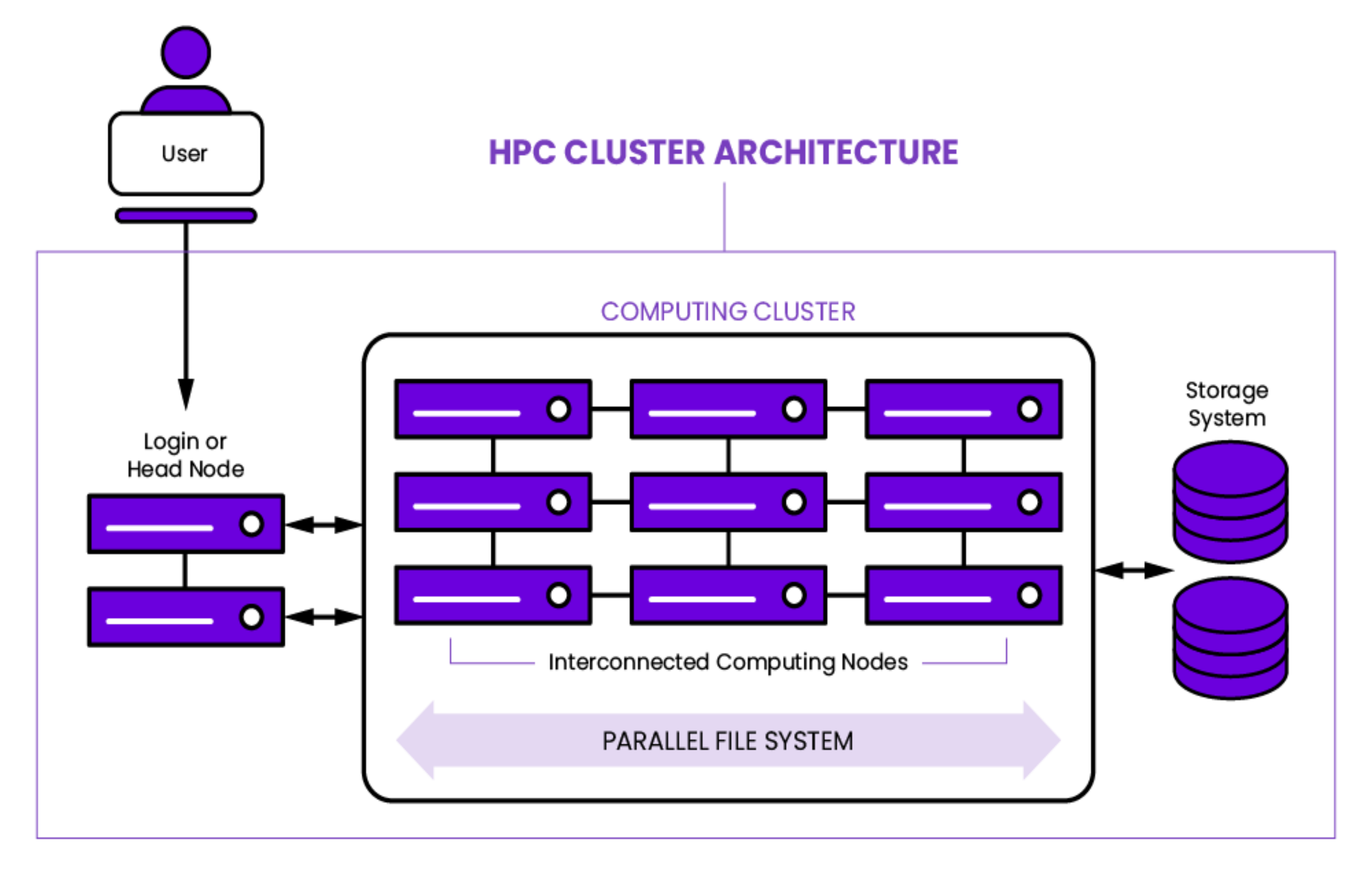
Temple University

HPC
High-Performance Computing

The Abdus Salam
International Centre
for Theoretical Physics

College of Science
and Technology

HPC Hardware



- **Hardware**
  - Compute Power
  - Network
  - File system
- **Software**
  - Deployment
  - Scheduler
  - User Software
- **Datacenter**
  - Cabinets
  - Power
  - Cooling
- **Total Cost of Ownership**

HPC Hardware

- Check if you are interested in learning from hardware basics to configure a cluster!

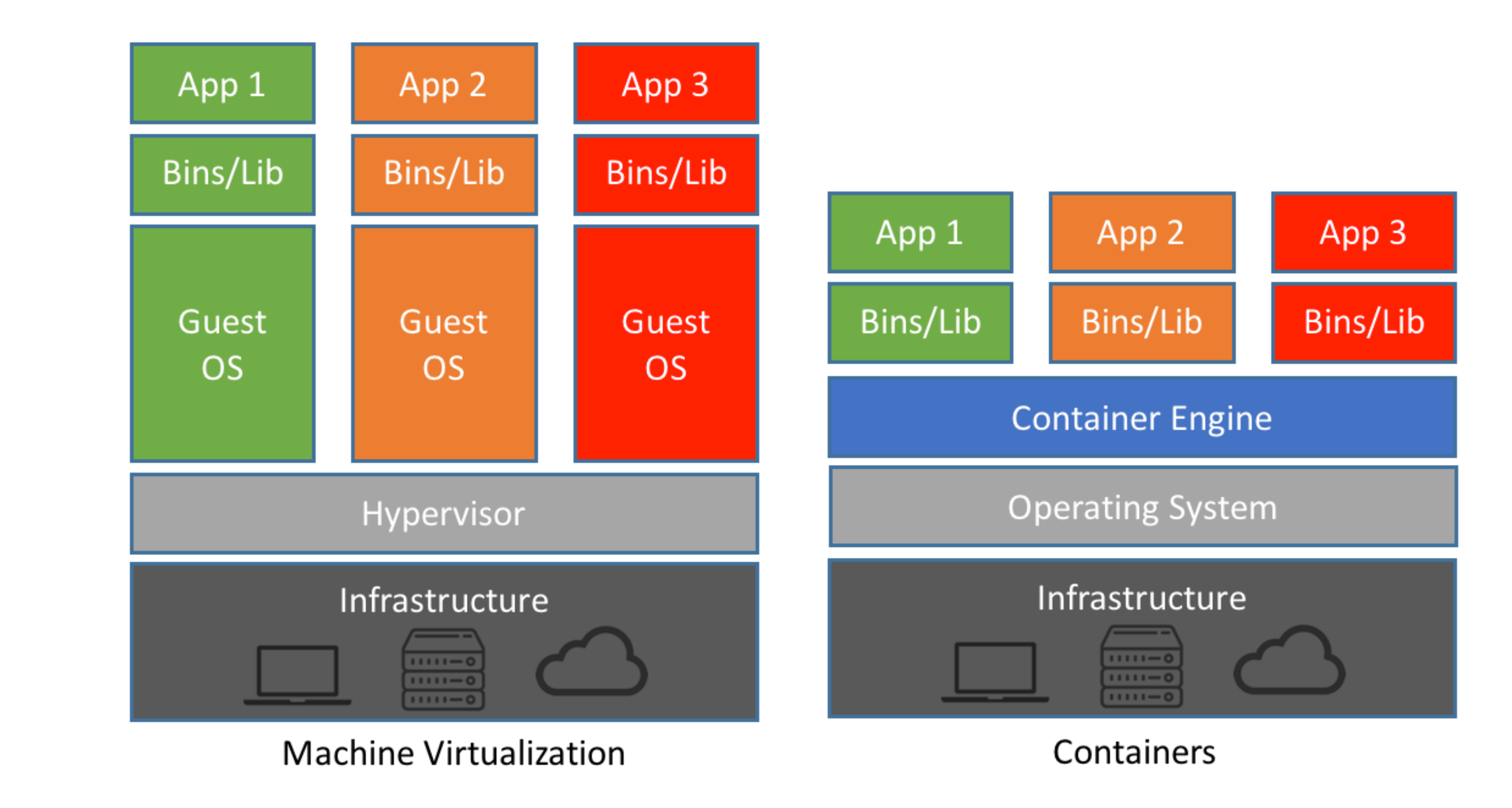
https://www.hpc.temple.edu/mhpc/hpc-technology/index.html

What is a Container?

- A container is an entity providing an isolated software environment (or filesystem) for an application and its dependencies.

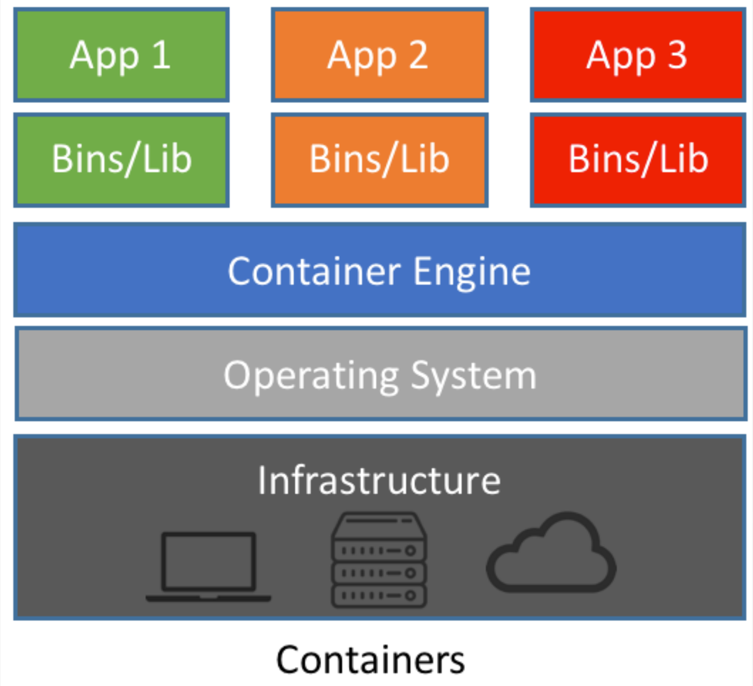
- Similar functional speaking to VMs like VirtualBox, Parallels, etc.

Container vs Virtual Machine (VM)



- VMs virtualize **Hardware**
- Containers virtualize **Operative Systems**

Container vs VM



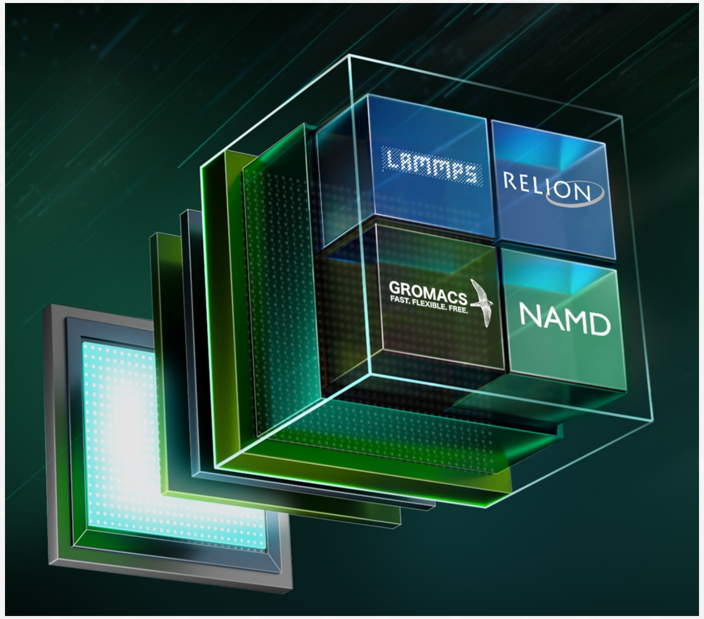
- Lighter weight to run (less CPU and memory usage, faster start-up times)
- Smaller in size (thus easier to transfer and share)
- Modular (possible to combine multiple containers that work together)

Why are Containers Important?

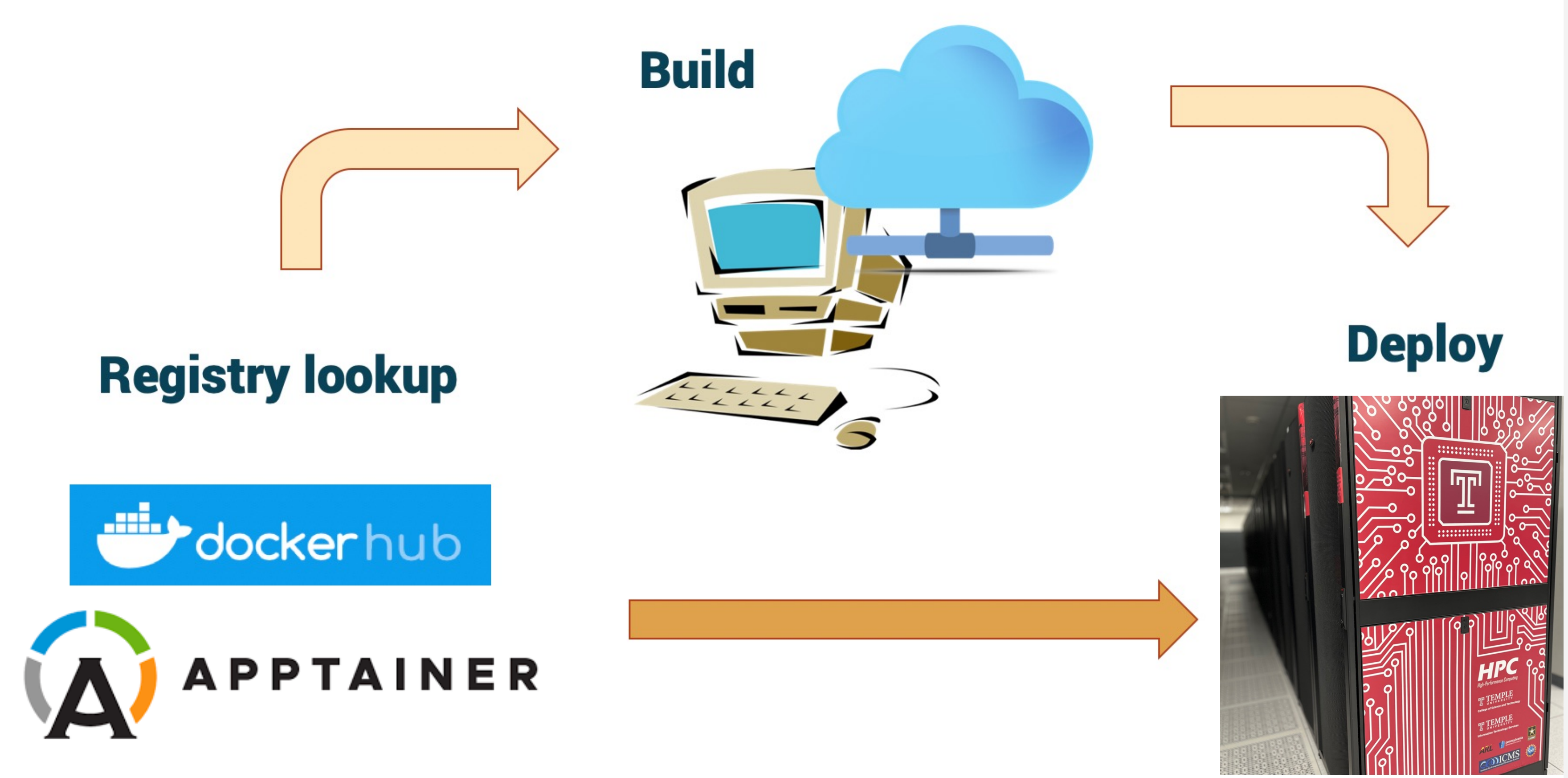
- Data reproducibility!
- Cross-system portability
- Simplified collaboration
- Simplified software dependencies and management
- Consistent testing environment

Workflows Using Containers

- Bioinformatics workflows
- Machine Learning
- RStudio & Jupyter Notebook
- Webservers
- Open Foam simulations
- Cloud workflows (via Singularity or Docker)
- HPC workflows (via Singularity)



Typical Workflow to use Containers



Terminology

Image

- Is a file (or set of files) that contains the application and all its dependencies, libraries, run-time systems, etc. required to run.

Container

- Is an instantiation of an image. That is, it’s a process in execution that got spawned out of an image. You can run multiple containers from the same image.

Registry

- Is a server application where images are stored and can be accessed by users

Recipe

- Is the definition File. ***def* file**, in Singularity (Apptainer) and ***Dockerfile*** in the Docker.

Container Engines

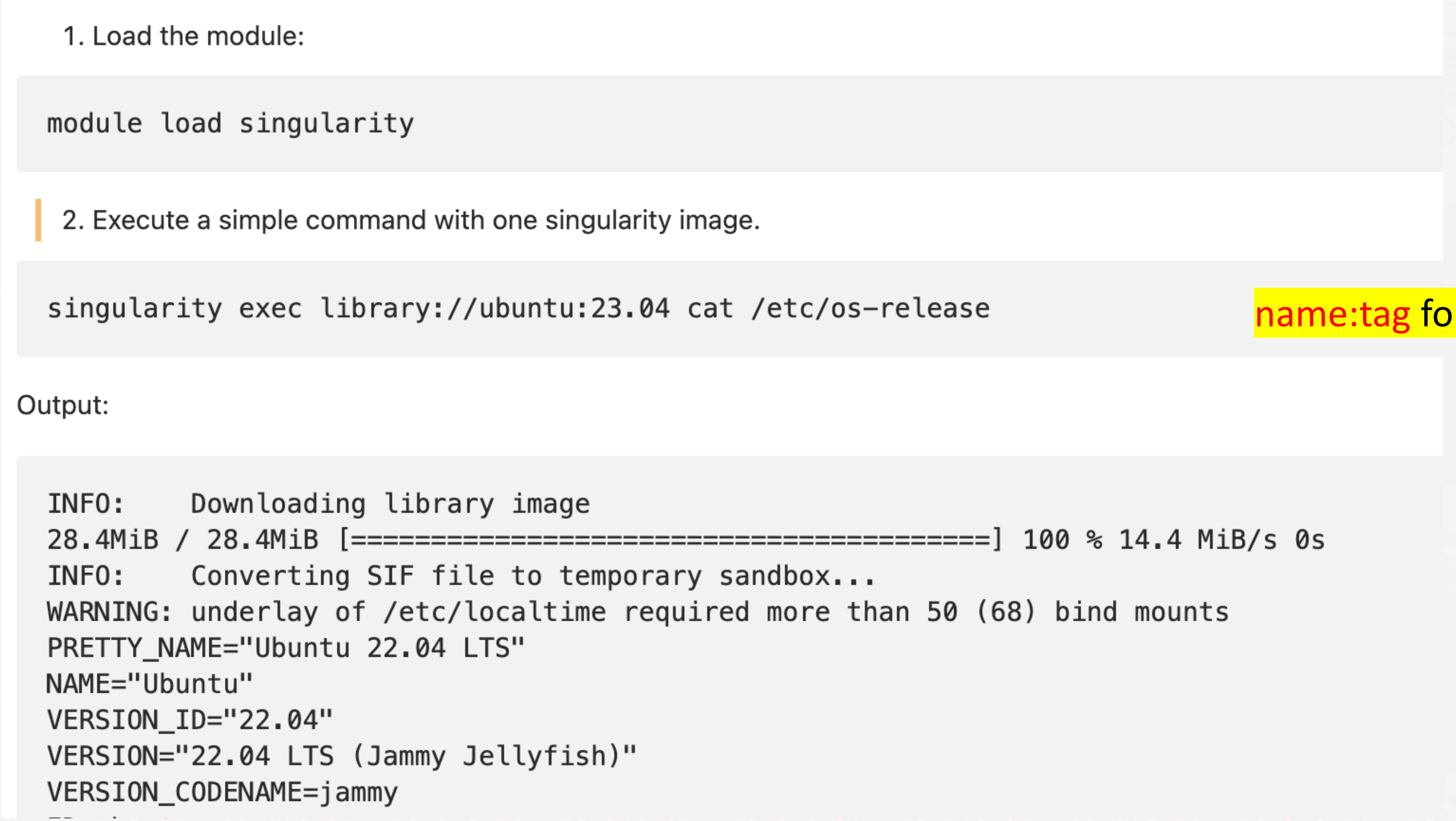
- **Docker**: Not very suitable for HPC as it requires root privileges to run.
- **Singularity**: a simple, powerful root-less container engine for the HPC world.
- **Apptainer**: an open-source offshoot of Singularity. Provides all the same functionality as Singularity and moving forward will likely become the open-source standard.

Singularity (Apptainer)

Singularity was designed from scratch as a container engine for HPC applications, which is clearly reflected in some of its main features:

- ***unprivileged* runtime**: Singularity containers do not require the user to hold root privileges to run
- ***integration***, rather than *isolation*, by default: same user as host, same shell variables inherited by host, current directory bind-mounted, communication ports available.
- Interface with job schedulers, such as *SLURM* or *PBS*;
- Ability to run MPI-enabled containers using host libraries;
- Native execution of GPU-enabled containers;
- Unfortunately, ***root* privileges are required to build container images.**

Simplest example



1. Load the module:

```
module load singularity
```

> 2. Execute a simple command with one singularity image.

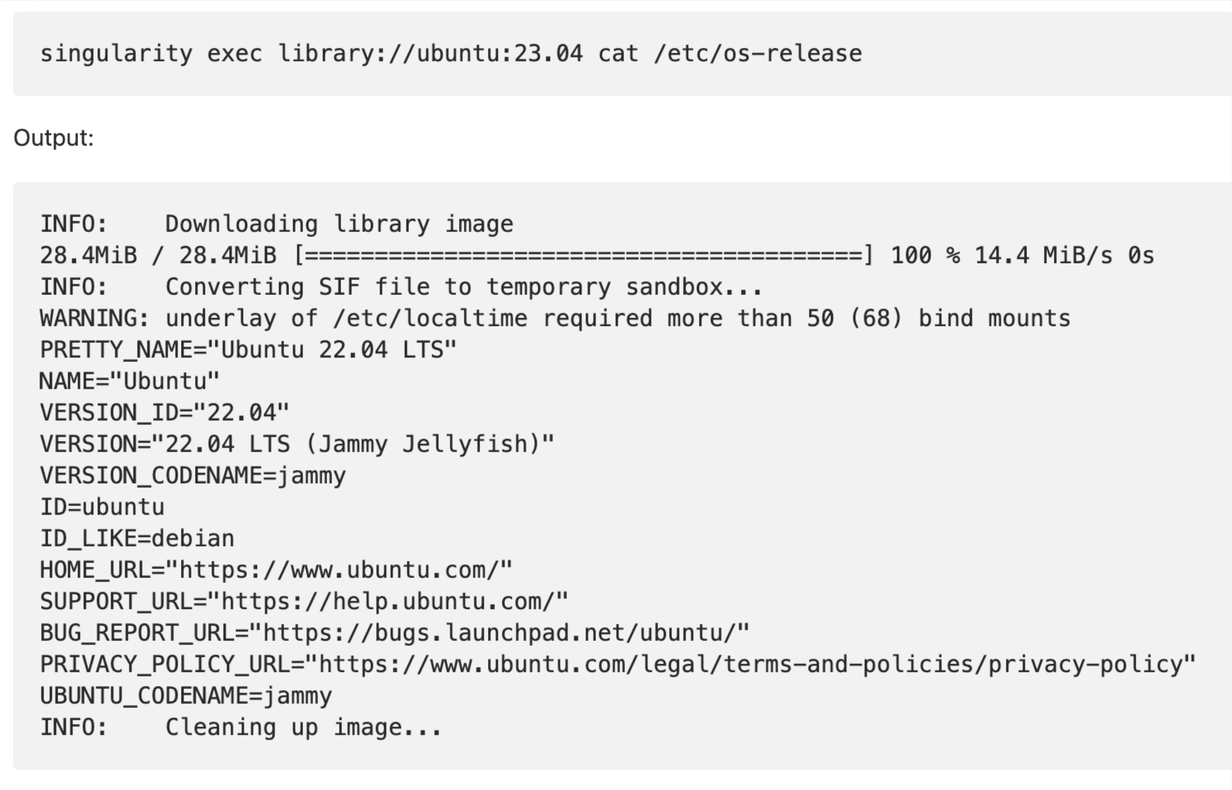
```
singularity exec library://ubuntu:23.04 cat /etc/os-release
```

Output:

```
INFO:    Downloading library image
28.4MiB / 28.4MiB [========================================] 100 % 14.4 MiB/s 0s
INFO:    Converting SIF file to temporary sandbox...
WARNING: underlay of /etc/localtime required more than 50 (68) bind mounts
PRETTY_NAME="Ubuntu 22.04 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
```

name:tag format for images.

Simplest example



```
singularity exec library://ubuntu:23.04 cat /etc/os-release
```

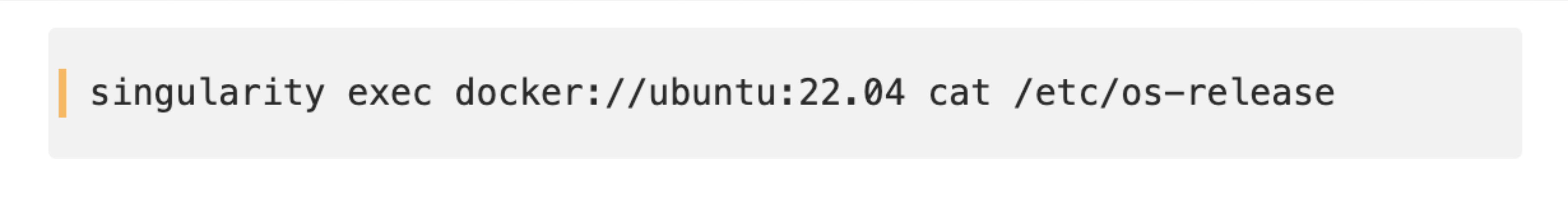
Output:

```
INFO:    Downloading library image
28.4MiB / 28.4MiB [========================================] 100 % 14.4 MiB/s 0s
INFO:    Converting SIF file to temporary sandbox...
WARNING: underlay of /etc/localtime required more than 50 (68) bind mounts
PRETTY_NAME="Ubuntu 22.04 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
INFO:    Cleaning up image...
```

This is what Singularity has done:

1. Downloaded an Ubuntu image from the Cloud Library (skipped if the image is already downloaded).
2. Stored it into the default cache directory.
3. Instantiated a container from the image.
4. executed the command.

Importing docker images with SI



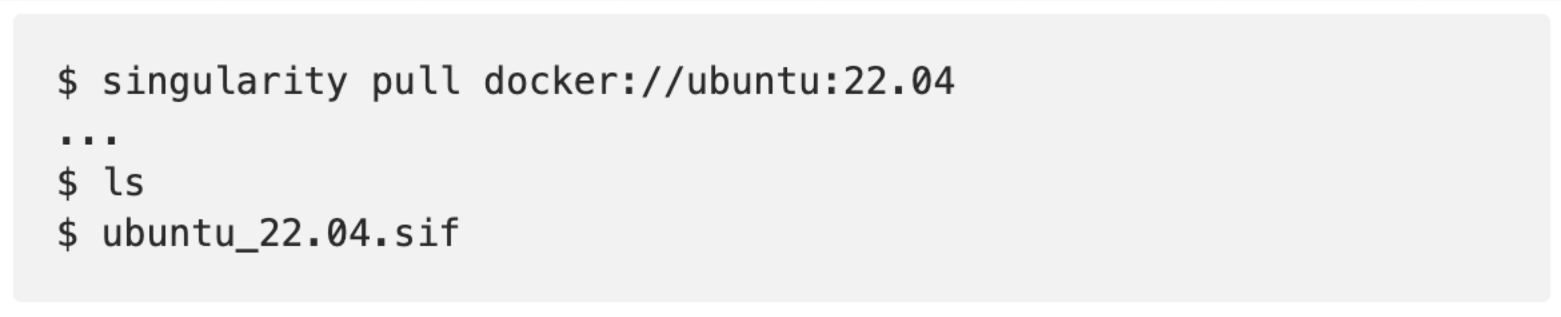
```
singularity exec docker://ubuntu:22.04 cat /etc/os-release
```

Rather than just downloading a SIF file, now there’s more work for Singularity, as it must:

- download the various layers making up the image, and
- assemble them into a single SIF image file.

Note that, to point Singularity to Docker Hub, the prefix docker:// is required.

Downloading images



```
$ singularity pull docker://ubuntu:22.04
...
$ ls
$ ubuntu_22.04.sif
```

Continue with the instructions on the tutorial, step 3…

Popular registries (aka image libraries)

- Bioinformatics: quay.io, biocontainers.pro
- NVIDIA: ngc.nvidia.com
- AMD: AMD Infinity Hub
- Singularity: cloud.sylabs.io
- Docker: http://hub.docker.com/

Building Containers

- Back to the tutorial…