

# An Introduction to Singularity: Containers for Scientific and High-Performance Computing

Marty Kandes, Ph.D.

High-Performance Computing User Services Group  
San Diego Supercomputer Center  
University of California, San Diego

SDSC Summer Institute 2020  
Friday, August 7th, 2020  
8:30AM - 9:00PM PDT

# Today's Session

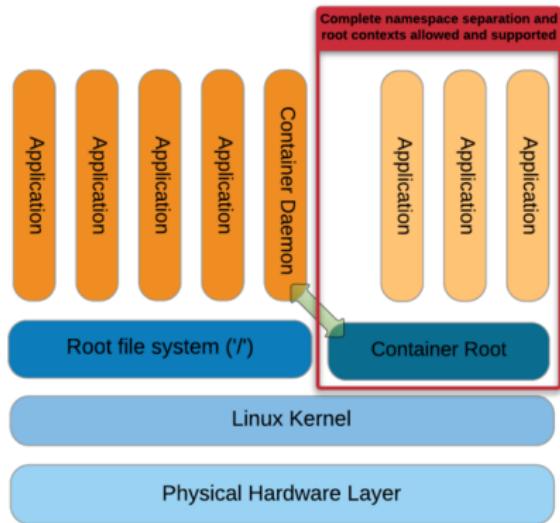
- ▶ What are containers? And why you should use them.
- ▶ What is Singularity and what does it have to do with containers?
- ▶ How to build and run Singularity containers.
- ▶ Q&A?

# Containers



# What is a Container?

- ▶ At rest, a container or **container image** is simply a file (or collection of files) saved on disk that stores everything you need to run a target application or applications: code, runtime, system tools, libraries and settings, etc.
- ▶ *In motion*, a container or **container process** is simply a standard (Linux) process running on top of the underlying host's operating system and kernel, but whose software environment is defined by the contents of the container image.



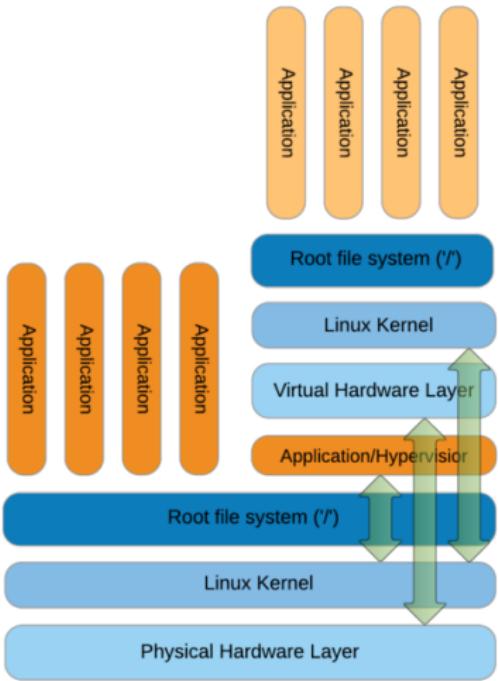
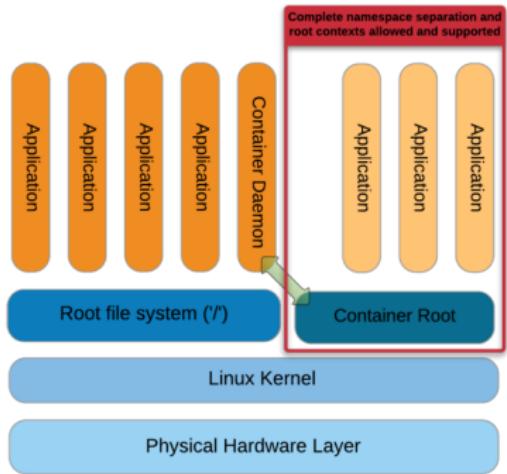
Container : Supercomputer :: Construct : Matrix

“ ... it's our loading program.”



“ We can load anything ... anything we need.”

# Containers vs. Virtual Machines



Container-based applications have **direct access** to the host kernel and hardware, *similar to native applications*. In contrast, VM-based applications only have **indirect access** via the guest OS and hypervisor, which creates a significant performance overhead.

# Advantages of Containers

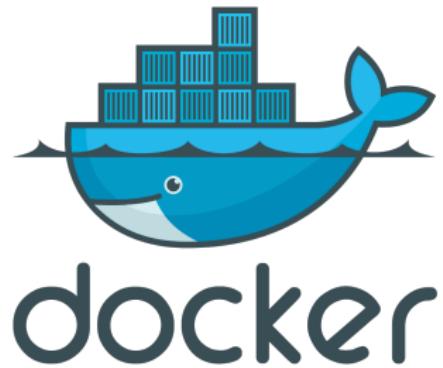
- ▶ **Performance:** Near-native application performance
- ▶ **Freedom:** Bring your own software environment
- ▶ **Reproducibility:** Package complex software applications into easy to manage, verifiable software units
- ▶ **Compatibility:** Built on open standards available in all major Linux distributions
- ▶ **Portability:** Build once, run (almost) anywhere

# Limitations of Containers

- ▶ **Architecture-dependent:** Always limited by CPU architecture (x86\_64, ARM) and binary format (ELF)
- ▶ **Portability:** Requires glibc and kernel compatibility between host and container; also requires any other kernel-user space API compatibility (e.g., OFED/IB, NVIDIA/GPUs)
- ▶ **Filesystem isolation:** filesystem paths are (mostly) different when viewed inside and outside container

# Docker

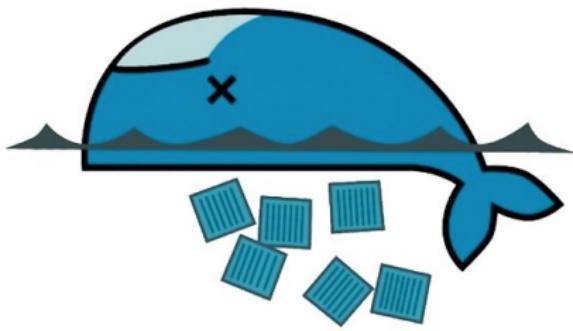
- ▶ Most common **container engine** in use today
- ▶ Provides tools and utilities to create, maintain, distribute, and run containers images
- ▶ Designed to accommodate network-centric services (web servers, databases, etc)
- ▶ Easy to install, well-documented, and large, well-developed user community and container ecosystem (DockerHub)



<https://www.docker.com>

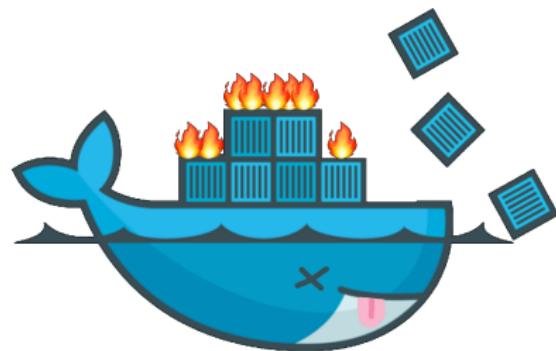
# Docker on HPC Systems

- ▶ HPC systems are shared resources
- ▶ Docker's security model is designed to support trusted users running trusted containers; e.g., users can escalate to root
- ▶ Docker not designed to support batch-based workflows
- ▶ Docker not designed to support tightly-coupled, highly distributed parallel applications (MPI).



## Docker on the Cloud (CVE-2019-5736)

- ▶ “The Open Containers Initiative (OCI) recently discovered a new security vulnerability CVE-2019-5736 in runc, allowing container escape to obtain root privileges on the host node.”
- ▶ “ Amazon employee here: we have released a security bulletin covering how to update to the latest patched Docker on Amazon Linux, Amazon ECS, Amazon EKS, AWS Fargate, AWS IoT Greengrass, AWS Batch, AWS Elastic Beanstalk, AWS Cloud9, AWS SageMaker, AWS RoboMaker, and AWS Deep Learning AMI.”



# Singularity: A Container Engine for HPC

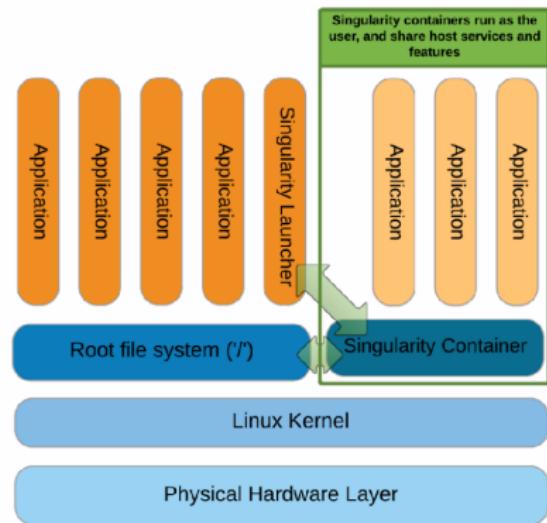
- ▶ Reproducible, portable, sharable, and distributable containers
- ▶ No trust security model: untrusted users running untrusted containers
- ▶ Support HPC hardware and scientific applications



<http://singularity.lbl.gov>  
<https://www.sylabs.io>

# Features of Singularity

- ▶ Each container is a single image file
- ▶ No root owned daemon processes
- ▶ No user contextual changes or root escalation allowed; user inside container is always the same user who started the container
- ▶ Supports shared/multi-tenant resource environments
- ▶ Supports HPC hardware: Infiniband, GPUs
- ▶ Supports HPC applications: MPI



# Common Singularity Use Cases

- ▶ Building and running applications that require newer system libraries than are available on host system
- ▶ Running commercial applications binaries that have specific OS requirements not met by host system
- ▶ Converting Docker containers to Singularity containers

# The Singularity Workflow

1. **Build** your Singularity containers on a local system where you have root or sudo access; e.g., a personal computer where you have installed Singularity
2. **Transfer** your Singularity containers to the HPC system where you want to run them
3. **Run** your Singularity containers on that HPC system



# Running Singularity on Mac OS X\*\* or Windows

1. Install VirtualBox on your personal computer:  
<https://www.virtualbox.org>
2. Create either an Ubuntu or Fedora-based virtual machine,  
where you will build and test your Singularity containers
3. Install Singularity\* on that virtual machine:  
<https://sylabs.io/guides/3.6/admin-guide/installation.html>

\* Recommendation: Install the same version of Singularity used on the HPC system where you plan to run your containers. If you plan to run on multiple HPC systems, then install the lowest version number you expect to use.

\*\* Beta Release: <https://www.sylabs.io/singularity-desktop-macos>

# Essential Singularity

The main Singularity command

```
singularity [options] <subcommand> [subcommand options] ...
```

has three essential subcommands:

- ▶ **build**: Build your own container from scratch using a Singularity definition (or recipe) file; download and assemble any existing Singularity container; or convert your containers from one format to another (e.g., from Docker to Singularity)
- ▶ **shell**: Spawn an interactive shell session in your container.
- ▶ **exec**: Execute an arbitrary command within your container.

# How to *build* a Singularity container ...

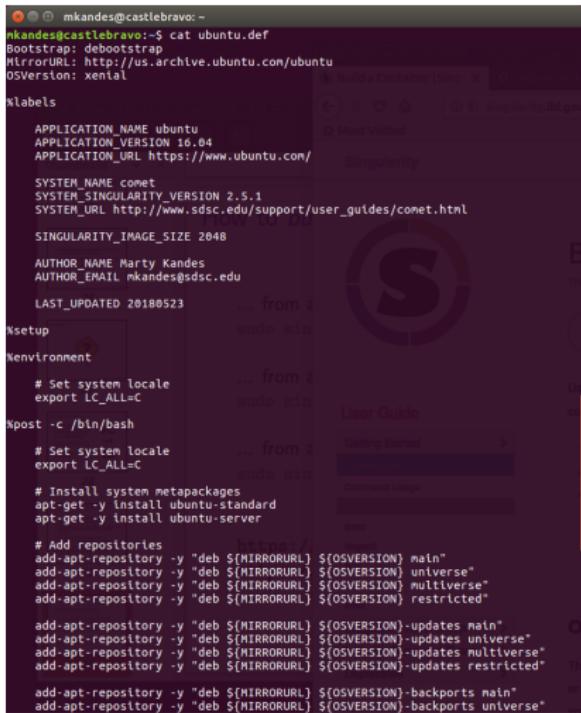
... from a Singularity definition (or recipe) file:

```
mkandes@castlebravo:~$ ls
Desktop Downloads Dropbox ubuntu.def
mkandes@castlebravo:~$ sudo singularity build ubuntu.simg ubuntu.def
Using container recipe deffile: ubuntu.def
Sanitizing environment
Adding base Singularity environment to container
I: Retrieving InRelease
I: Checking Release signature
I: Valid Release signature (key id 790BC7277767219C42C86F933B4FE6ACC0B21F32)
I: Retrieving Packages
I: Validating Packages
I: Resolving dependencies of required packages...
I: Resolving dependencies of base packages...
I: Found additional base dependencies: gcc-5-base gnupg gpgv libapt-pkg5.0 liblzl
4-1 libreadline6 libstdc++6 libusb-0.1-4 readline-common ubuntu-keyring
I: Checking component main on http://us.archive.ubuntu.com/ubuntu...
I: Retrieving adduser 3.113+nmu3ubuntu4
I: Validating adduser 3.113+nmu3ubuntu4
I: Retrieving apt 1.2.10ubuntu1
I: Validating apt 1.2.10ubuntu1
I: Retrieving base-files 9.4ubuntu4
I: Validating base-files 9.4ubuntu4
I: Retrieving base-passwd 3.5.39
I: Validating base-passwd 3.5.39
```

```
sudo singularity build ubuntu.sif ubuntu.def
```

# Singularity Recipe File

- ▶ A Singularity definition (or recipe) file is the starting point for designing any custom container.
- ▶ It is a manifest of all software to be installed within the container, environment variables to be set, files to be added, directories to be mounted, container metadata, etc.
- ▶ You can even write a help section, or define modular components in the container.



A screenshot of a terminal window titled "mkandes@castlebravo: ~". The window displays a Singularity recipe file named "ubuntu.def". The file contains the following content:

```
Bootstrap: debootstrap
MirrorURL: http://us.archive.ubuntu.com/ubuntu
OSVersion: xenial

LABELS
APPLICATION_NAME ubuntu
APPLICATION_VERSION 16.04
APPLICATION_URL https://www.ubuntu.com/

SYSTEM_NAME comet
SYSTEM_SINGULARITY_VERSION 2.5.1
SYSTEM_URL http://www.sdsc.edu/support/user_guides/comet.html

SINGULARITY_IMAGE_SIZE 2048B

AUTHOR_NAME Marty Kandes
AUTHOR_EMAIL mkandes@sdsc.edu

LAST_UPDATED 20180523
```

The terminal window also shows a sidebar with links like "Setup", "Environment", "User Guide", "Getting Started", "Command Usage", and "About".

## naked-singularity

- ▶ A repository of definition (or recipe) files for building Singularity containers around the software applications, frameworks, and libraries you need to run on high-performance computing systems.
- ▶ Aim of the project is to:
  1. Version control the Singularity containers we're building, maintaining, and deploying for you;
  2. Make it easy for you to see what is installed within these Singularity containers; and
  3. Make available to you the same base definition files we use to build our Singularity containers, which can serve as a starting point for your own custom Singularity containers.
- ▶ <https://github.com/mkandes/naked-singularity>

# How to *build* a Singularity container ...

... from a Singularity definition (or recipe) file:

```
mkanedes@castlebravo:~$ sudo singularity build ubuntu.sif ubuntu.def
installing: ruamel.yaml-0.15.46-py37h14c3975_0 ...
installing: six-1.11.0-py37_1 ...
installing: cffi-1.11.5-py37he75722e_1 ...
installing: setuptools-40.2.0-py37_0 ...
installing: cryptography-2.3.1-py37hc365091_0 ...
installing: wheel-0.31.1-py37_0 ...
installing: pip-10.0.1-py37_0 ...
installing: pyopenssl-18.0.0-py37_0 ...
installing: urllib3-1.23-py37_0 ...
installing: requests-2.19.1-py37_0 ...
installing: conda-4.5.11-py37_0 ...
installation finished.
Adding deffile section labels to container
Adding runscript
Running test scriptlet
Finalizing Singularity container
Calculating final size for metadata...
Skipping checks
Building Singularity image...
Singularity container built: ubuntu.simg
Cleaning up...
mkanedes@castlebravo:~$ ls
Desktop  Downloads  Dropbox  ubuntu.def  ubuntu.simg
mkanedes@castlebravo:~$
```

```
sudo singularity build ubuntu.sif ubuntu.def
```

Docker Hub - Mozilla Firefox

Docker Hub    Singularity Hub

https://hub.docker.com

Search

Explore Help Sign in

# Docker Hub

Dev-test pipeline automation, 100,000+ free apps, public and private registries

New to Docker?  
Create your free Docker ID to get started.

Choose a Docker ID

Email address

Choose a password

I agree to Docker's [Terms of Service](#).  
 I agree to Docker's [Privacy Policy and Data Processing Terms](#).  
 I would like to receive email updates from Docker, including its various services and products

Sign Up

© 2016 Docker Inc.

<https://hub.docker.com>

# How to *build* a Singularity container ...

... from an existing Docker container on DockerHub:

```
mkandes@castlebravo:~$ sudo singularity build ubuntu-docker.simg docker://ubuntu
Docker image path: index.docker.io/library/ubuntu:latest
Cache folder set to /root/.singularity/docker
[5/5] ====== 100.0%
Importing: base Singularity environment
Exploding layer: sha256:124c757242f88002a858c23fc79f8262f9587fa30fd92507e586ad07
4afb42b6.tar.gz
Exploding layer: sha256:9d866f8bde2a0d607a6d17edc0fb5e00b58306efc2b0a57e0ba72f2
69e7c6be.tar.gz
Exploding layer: sha256:fa3f2f277e67c5ccb1dac21dc27111a60d3cd2ef494d94aa1515d33
19f2a245.tar.gz
Exploding layer: sha256:398d32b153e84fe343f0c5b07d65e89b05551aae6cb8b3a03bb2b662
976eb3b8.tar.gz
Exploding layer: sha256:afde35469481d2bc446d649a7a3d099147bbf7696b66333e76a41168
6b617ea1.tar.gz
Exploding layer: sha256:c6a9ef4b9995d615851d7786fbc2fe72f72321beela87d66919b881a
0336525a.tar.gz
Building Singularity image...
Singularity container built: ubuntu-docker.simg
Cleaning up...
mkandes@castlebravo:~$ ls
Desktop Downloads Dropbox ubuntu.def  ubuntu-docker.simg  ubuntu.simg
mkandes@castlebravo:~$
```

```
sudo singularity build ubuntu-docker.sif docker://ubuntu
```

Singularity Hub - Mozilla Firefox

Docker Hub Singularity Hub

https://www.singularity-hub.org

SINGULARITYHUB Collections About User Guide Tools Search Login

# Singularity Container Registry

A collaboration between [Stanford University](#) and [SingularityLLC](#)

## Singularity Hub 2.1

Hi friends! Welcome to Singularity Hub. Read the [Release Notes](#) for important changes to the platform,



<https://www.singularity-hub.org>

# How to *build* a Singularity container ...

... from an existing Singularity container on SingularityHub:

```
mkandes@castlebravo:~$ Exploding layer: sha256:398d32b153e84fe343f0c5b07d65e89b05551aae6cb8b3a03bb2b662  
976eb3b8.tar.gz  
Exploding layer: sha256:afde35469481d2bc446d649a7a3d099147bbf7696b66333e76a41168  
6b617ea1.tar.gz  
Exploding layer: sha256:c6a9ef4b9995d615851d7786fc2fe72f72321beela87d66919b881a  
0336525a.tar.gz  
Building Singularity image...  
Singularity container built: ubuntu-docker.simg  
Cleaning up...  
mkandes@castlebravo:~$ ls  
Desktop Downloads Dropbox ubuntu.def ubuntu-docker.simg ubuntu.simg  
mkandes@castlebravo:~$ sudo singularity build ubuntu-shub.simg shub://singularit  
yhub/ubuntu  
Cache folder set to /root/.singularity/shub  
Progress |=====| 100.0%  
Building from local image: /root/.singularity/shub/singularityhub-ubuntu-master-  
latest.simg  
Building Singularity image...  
Singularity container built: ubuntu-shub.simg  
Cleaning up...  
mkandes@castlebravo:~$ ls  
Desktop Dropbox ubuntu-docker.simg ubuntu.simg  
Downloads ubuntu.def ubuntu-shub.simg  
mkandes@castlebravo:~$
```

```
sudo singularity build ubuntu-shub.sif shub://mkandes/ubuntu
```

Sylabs Cloud - Mozilla Firefox

Sylabs Cloud https://cloud.sylabs.io/home

Sylabs Library Remote Builder Keystore Help Sign in to Sylabs



### Secure

Ensure authenticity and integrity with digital signatures.



### Create [alpha preview]

Build a Singularity container image in the cloud.



### Share [alpha preview]

Discover and publish container images with the world.



### Container Library

Container Library is the official image registry provided by Sylabs.io, users can share singularity images through the cloud library, user can also pull/push images through Singularity CLI.



### Remote Builder

Remote Builder allows user build Singularity image in Cloud directly, without install OS/Package locally, without give additional privilege

<https://cloud.sylabs.io>

If you *build* it ...



... what can you do with it?

# Essential Singularity

The main Singularity command

```
singularity [options] <subcommand> [subcommand options] ...
```

has three essential subcommands:

- ▶ **build**: Build your own container from scratch using a Singularity definition (or recipe) file; download and assemble any existing Singularity container; or convert your containers from one format to another (e.g., from Docker to Singularity)
- ▶ **shell**: Spawn an interactive shell session in your container.
- ▶ **exec**: Execute an arbitrary command within your container.

# How to spawn an interactive *shell* ...

... within a Singularity container:

```
mkandes@castlebravo:~$ ls
Desktop  Dropbox  ubuntu-docker.simg  ubuntu.simg
Downloads  ubuntu.def  ubuntu-shub.simg
mkandes@castlebravo:~$ cat /etc/*release | grep DISTRIB
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04.4 LTS"
mkandes@castlebravo:~$ singularity shell ubuntu-docker.simg
Singularity: Invoking an interactive shell within container...

Singularity ubuntu-docker.simg:~> cat /etc/*release | grep DISTRIB
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.1 LTS"
Singularity ubuntu-docker.simg:~> ls
Desktop  Dropbox  ubuntu-shub.simg  ubuntu.simg
Downloads  ubuntu-docker.simg  ubuntu.def
Singularity ubuntu-docker.simg:~> whoami
mkandes
Singularity ubuntu-docker.simg:~> exit
exit
mkandes@castlebravo:~$
```

`singularity shell ubuntu-docker.sif`

# Use the interactive *shell* to ...

... explore the software environment of a Singularity container:

```
mkandes@castlebravo:~$ ls
Desktop  Dropbox  ubuntu-docker.simg  ubuntu.simg
Downloads  ubuntu.def  ubuntu-shub.simg
mkandes@castlebravo:~$ which python
/usr/bin/python
mkandes@castlebravo:~$ python --version
Python 2.7.12
mkandes@castlebravo:~$ which python3
/usr/bin/python3
mkandes@castlebravo:~$ python3 --version
Python 3.5.2
mkandes@castlebravo:~$ singularity shell ubuntu-docker.simg
Singularity: Invoking an interactive shell within container...

Singularity ubuntu-docker.simg:~> which python
Singularity ubuntu-docker.simg:~> python --version
bash: python: command not found
Singularity ubuntu-docker.simg:~> which python3
Singularity ubuntu-docker.simg:~> python3 --version
bash: python3: command not found
Singularity ubuntu-docker.simg:~> exit
exit
mkandes@castlebravo:~$ █
```

`singularity shell ubuntu-docker.sif`

# Use the interactive *shell* to ...

... modify the contents of a Singularity container:

```
mkandes@castlebravo: ~
Downloads  ubuntu.def  ubuntu-shub.simg
mkandes@castlebravo:~$ which python
/usr/bin/python
mkandes@castlebravo:~$ python --version
Python 2.7.12
mkandes@castlebravo:~$ which python3
/usr/bin/python3
mkandes@castlebravo:~$ python3 --version
Python 3.5.2
mkandes@castlebravo:~$ singularity shell ubuntu-docker.simg
Singularity: Invoking an interactive shell within container...
Singularity ubuntu-docker.simg:~> which python
Singularity ubuntu-docker.simg:~> python --version
bash: python: command not found
Singularity ubuntu-docker.simg:~> which python3
Singularity ubuntu-docker.simg:~> python3 --version
bash: python3: command not found
Singularity ubuntu-docker.simg:~> exit
mkandes@castlebravo:~$ sudo singularity shell --writable ubuntu-docker.simg
ERROR  : Unable to open squashfs image in read-write mode: Read-only file system
ABORT  : Retval = 255
mkandes@castlebravo:~$
```

```
sudo singularity shell --writable ubuntu-docker.sif
```



```
ERROR : Unable to open squashfs image in read-write mode:  
        Read-only file system  
ABORT : Retval = 255
```

# Singularity Container Image Formats

There are now (only) 2 different Singularity container image formats:

- ▶ Compressed READ-ONLY **Singularity Image File (SIF)** format suitable for production (default)
- ▶ writable **(ch)root directory** called a sandbox for interactive (`--writable`) development (`-sandbox` option)

# How to build a --sandbox Singularity container ...

... from an existing Singularity container on SingularityHub:

```
mkandes@castlebravo:~/lolcow
mkandes@castlebravo:~$ sudo singularity build --sandbox lolcow shub://GodloveD/lolcow
[sudo] password for mkandes: It's good idea to do this as root to ensure you have permission to access
Cache folder set to /root/.singularity/shub
Progress |=====| 100.0%
Building from local image: /root/.singularity/shub/GodloveD-lolcow-master-latest.simg
mkandes@castlebravo:~$ singularity shell --writable lolcow/
Singularity container built: lolcow
Cleaning up...
mkandes@castlebravo:~$ singularity run lolcow/
/ Your reasoning powers are good, and you \
\ are a fairly good planner.

----- If you saved locally, you can use it as a target to build a new
\ \ ^ ^ containers from one format to another. For example if you had a squashfs container o
(oo)\_____
(_)\ )\ \\\\
 ||----w |
 ||      |
mkandes@castlebravo:~$ cd lolcow/
mkandes@castlebravo:~/lolcow$ ls
bin dev etc lib media opt root sbin srv tmp var
boot environment home lib64 mnt proc run singularity sys usr
mkandes@castlebravo:~/lolcow$
```

```
sudo singularity build --sandbox lolcow shub://GodloveD/lolcow
```

# Now What?!!



# Essential Singularity

The main Singularity command

```
singularity [options] <subcommand> [subcommand options] ...
```

has three essential subcommands:

- ▶ **build**: Build your own container from scratch using a Singularity definition (or recipe) file; download and assemble any existing Singularity container; or convert your containers from one format to another (e.g., from Docker to Singularity)
- ▶ **shell**: Spawn an interactive shell session in your container.
- ▶ **exec**: Execute an arbitrary command within your container.

## How to execute ...

... arbitrary commands within a Singularity container:

```
mkandes@castlebravo:~$ ls
Desktop  Dropbox  ubuntu-docker.img  ubuntu.img      ubuntu.simg
Downloads  ubuntu.def  ubuntu-docker.simg  ubuntu-shub.simg
mkandes@castlebravo:~$ python --version
Python 2.7.12
mkandes@castlebravo:~$ python3 --version
Python 3.5.2
mkandes@castlebravo:~$ singularity exec ubuntu-docker.img python --version
Python 2.7.15rc1
mkandes@castlebravo:~$ singularity exec ubuntu-docker.img python3 --version
/.singularity.d/actions/exec: 9: exec: python3: not found
mkandes@castlebravo:~$
```

```
singularity exec ubuntu-docker.img python --version
```

# Essential Singularity

The main Singularity command

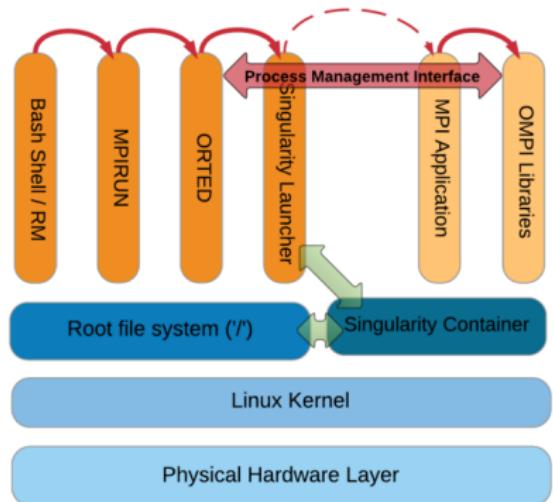
```
singularity [options] <subcommand> [subcommand options] ...
```

has three essential subcommands:

- ▶ `build`: Build your own container from scratch using a Singularity definition (or recipe) file; download and assemble any existing Singularity container; or convert your containers from one format to another (e.g., from Docker to Singularity)
- ▶ `shell`: Spawn an interactive shell session in your container.
- ▶ `exec`: Execute an arbitrary command within your container.

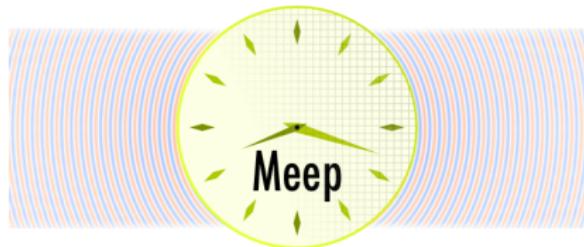
# MPI-based Singularity Containers

- ▶ Use same Message Passing Interface (MPI) distribution and version within container as would be used outside the container.
- ▶ If using Infiniband (IB), install same OFED drivers and libraries inside the container as used on underlying HPC hardware.



# MPI-based Singularity Containers: MEEP

- ▶ MEEP: MIT Electromagnetic Equation Propagation is a free and open-source software package for simulating electromagnetic systems via the finite-difference time-domain (FDTD) method.
- ▶ Dependency hell: Too difficult to compile in TSCC's native software environment.



# MPI-based Singularity Containers: MEEP

```
mkanedes@comet-ln2:~/Software/meep
#!/usr/bin/env bash

#SBATCH --job-name="meep-example"
#SBATCH --account=use300
#SBATCH --partition=debug
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=24
#SBATCH --time=00:30:00
#SBATCH --no-requeue
#SBATCH --output="meep-example.o%j.%N"

declare -xr COMPILER_MODULE='gnu/4.9.2'
declare -xr MPI_MODULE='openmpi_ib/1.8.4'
declare -xr SINGULARITY_MODULE='singularity/2.6.1'

declare -xr SINGULARITY_IMAGE_DIR='/share/apps/compute/singularity/images'

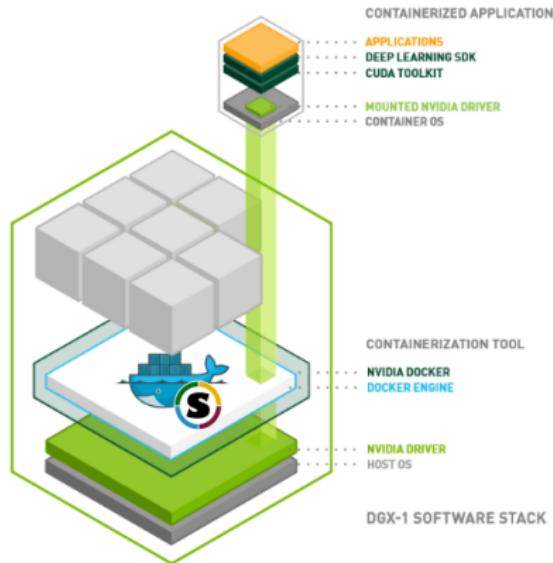
module purge
module load "${COMPILER_MODULE}"
module load "${MPI_MODULE}"
module load "${SINGULARITY_MODULE}"
module list
printenv

time -p ibrun singularity exec "${SINGULARITY_IMAGE_DIR}/meep/meep.simg" \
    meep /opt/meep/scheme/examples/parallel-wvgs-force.ctl
1,2          All
```

```
mpirun -np X singularity exec meep.sif meep parallel-wvgs-force.ctl
```

# GPU-accelerated Singularity Containers

- ▶ GPU-accelerated containers also require an interface for accessing GPU drivers and libraries on the underlying host system.
- ▶ Traditionally, you would install the same driver and libraries within container that match distribution and version of them available on the host system.
- ▶ Today, Singularity actually allows you to bind mount the GPU driver and its supporting libraries at runtime with the `--nv` option.



# GPU-accelerated Singularity Containers: TensorFlow

- ▶ TensorFlow is an open source software library for high performance numeric and symbolic computation, and is most popularly used today for machine learning applications such as neural networks.
- ▶ Like many of the most popular machine learning frameworks, TensorFlow continues to evolve rapidly. At present, the latest versions of TensorFlow are incompatible with the version of TSCC's glibc library.



# GPU-accelerated Singularity Containers: TensorFlow

```
mkandes@comet-ln3:~/Software/tensorflow
#!/usr/bin/env bash

#SBATCH --job-name="tensorflow-gpu-example"
#SBATCH --account=use300
#SBATCH --partition=gpu-shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=6
#SBATCH --gres=gpu:1
#SBATCH --time=00:30:00
#SBATCH --no-requeue
#SBATCH --output="tensorflow-gpu-example.o%j.%N"

declare -xr SINGULARITY_MODULE='singularity/2.6.1'
declare -xr SINGULARITY_IMAGE_DIR='/share/apps/gpu/singularity/images'

module purge
module "${SINGULARITY_MODULE}"
module list
printenv

time -p singularity exec "${SINGULARITY_IMAGE_DIR}/tensorflow/tensorflow-gpu.simg" \
    python2 /opt/tensorflow/tensorflow/examples/tutorials/mnist/mnist_deep.py

time -p singularity exec "${SINGULARITY_IMAGE_DIR}/tensorflow/tensorflow-gpu.simg" \
    python3 /opt/tensorflow/tensorflow/examples/tutorials/mnist/mnist_deep.py
-
```

14,50 All

```
singularity exec keras-tensorflow-gpu.sif python mnist_deep.py
```

# GPU-accelerated Singularity Containers: TensorFlow

```
mkandes@comet-ln3:~/Software/tensorflow
#!/usr/bin/env bash

#SBATCH --job-name="tensorflow-gpu-example"
#SBATCH --account=use300
#SBATCH --partition=gpu-shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=6
#SBATCH --gres=gpu:1
#SBATCH --time=00:30:00
#SBATCH --no-requeue
#SBATCH --output="tensorflow-gpu-example.o%j.%N"

declare -xr SINGULARITY_MODULE='singularity/2.6.1'
declare -xr SINGULARITY_IMAGE_DIR='/share/apps/gpu/singularity/images'

module purge
module "${SINGULARITY_MODULE}"
module list
printenv

time -p singularity exec --nv "${SINGULARITY_IMAGE_DIR}/tensorflow/tensorflow-gpu.simg" \
    python2 /opt/tensorflow/tensorflow/examples/tutorials/mnist/mnist_deep.py

time -p singularity exec --nv "${SINGULARITY_IMAGE_DIR}/tensorflow/tensorflow-gpu.simg" \
    python3 /opt/tensorflow/tensorflow/examples/tutorials/mnist/mnist_deep.py
-
"run-tensorflow-gpu.slurm" 25L, 805C written
12,0-1          All
```

```
singularity exec --nv keras-tensorflow-gpu.sif python mnist_deep.py
```

## Exoskeletal (Dependency-Only) Singularity Containers

- ▶ Some software applications may need to be installed in your \$HOME directory on the underlying host system, but may require software dependencies not available and/or not supported on the host system. e.g., Julia with GPU support.
- ▶ Most often, users simply want to install additional packages in the Singularity containers we build and maintain for you.
- ▶ In both cases, the you can use any container's environment to install the software you need to in your \$HOME directory. Note, however, this can create some dependency confusion down the road when you forget which packages were installed with what container.

```
singularity exec ubuntu.sif pip install --user networkx
```

# Singularity: A Summary

1. You can now install (almost) any software you like on your favorite HPC system without having to make a special request to the system's administrators or user support staff.
2. In many cases, your software is now completely portable between the different HPC systems you want to run on.
3. And finally, you now have discrete software units (containers) that you can use to help maintain science reproducibility over the lifetime of a project, independent of how the software environment on any given HPC system changes over time.

# Questions?

