Introduction to Singularity: A Container Platform for Scientific and High-Performance Computing

Marty Kandes, Ph.D.

Computational & Data Science Research Specialist
High-Performance Computing User Services Group
San Diego Supercomputer Center
University of California, San Diego

CIML Summer Institute
Tuesday, June 22nd, 2021
12:25 PM - 1:05 PM PST

Today

- What is a container?
- What is Singularity?
- What can Singularity do for you?

Shipping Container

Est. 1956

Before Containerization (Bulk Break Cargo)

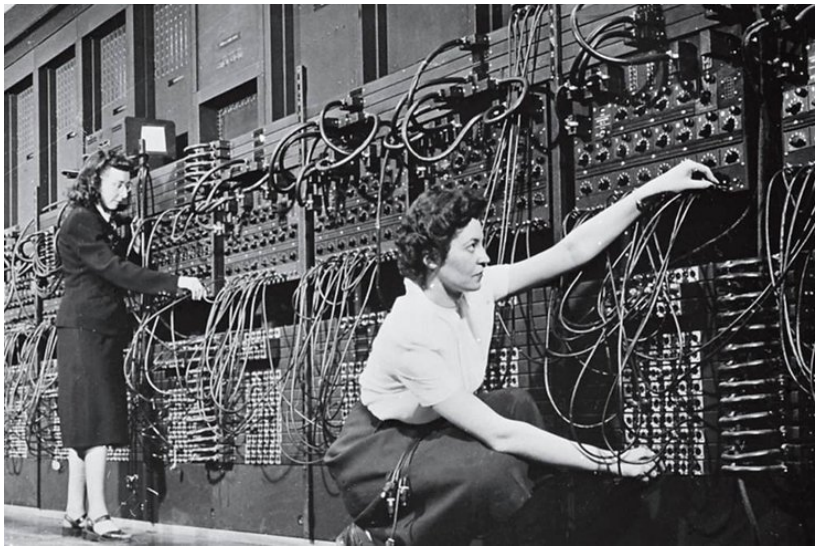
- Bespoke - goods must be loaded and unloaded individually, many times by hand
- Inefficient - more time spent loading and unloading goods than transporting them
- Insecure - goods must be stored and handled by intermediaries during transport; potential loss and/or theft of goods
- Local - only luxury and specialty goods shipped long-distance

After Containerization (Intermodal Freight Transport)

- Standardized - containers of known dimensions and permissible weight tolerances
- Efficient - portable containers allow fast loading and unloading from multiple modes of transportation
- Secure - goods remained stored within the same container during transport
- Global - cost effective to ship almost any good from anywhere in the world

Computing ≈ Shipping?

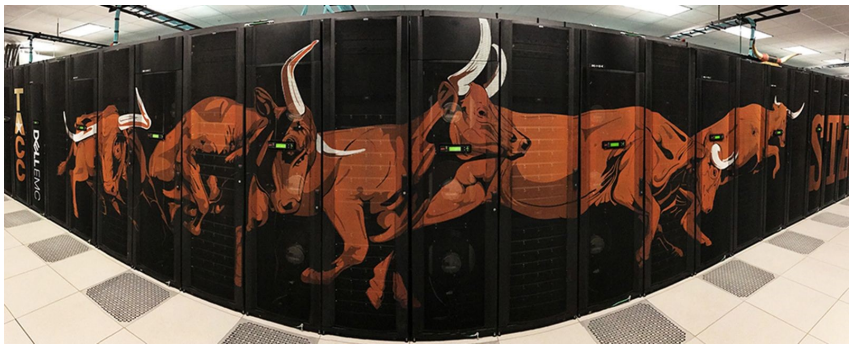
Evolution of Software Deployment

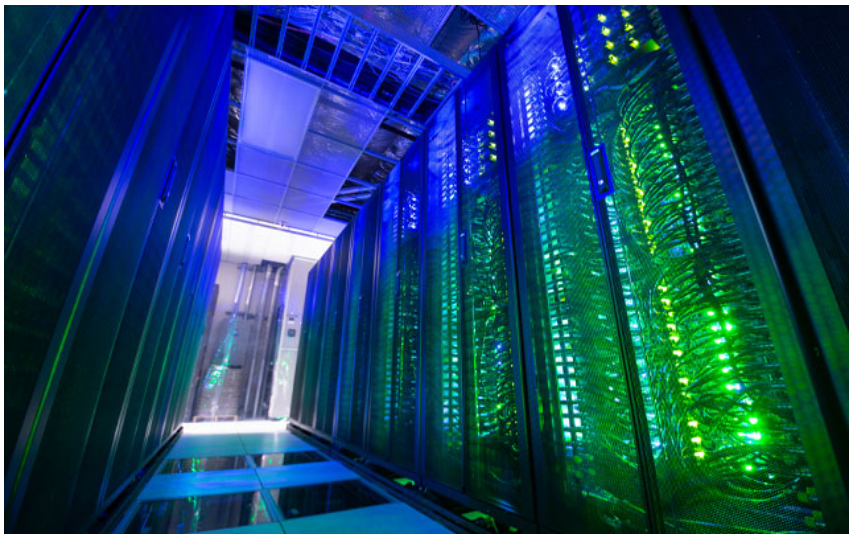


~ (less)

```
saws> aws ec2 help
EC2()                                                    EC2()



NAME
       ec2 -

DESCRIPTION
       Amazon  Elastic Compute Cloud (Amazon EC2) provides resizable computing
       capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 elim-
       inates your need to invest in hardware up front, so you can develop and
       deploy applications faster.

AVAILABLE COMMANDS
       o accept-vpc-peering-connection

       o allocate-address

       o assign-private-ip-addresses

       o associate-address

       o associate-dhcp-options

       o associate-route-table
```

Software Deployment on XSEDE





Comet vs. Stampede2

mkandes — mkandes@comet-ln3:~/gpse — ssh comet — 80×43

```
[mkandes@comet-ln3 ~]$ git clone https://github.com/mkandes/gpse.git
Cloning into 'gpse'...
remote: Enumerating objects: 528, done.
remote: Total 528 (delta 0), reused 0 (delta 0), pack-reused 528
Receiving objects: 100% (528/528), 311.06 KiB | 1.31 MiB/s, done.
Resolving deltas: 100% (347/347), done.
Checking out files: 100% (15/15), done.
[mkandes@comet-ln3 ~]$ cd gpse/
[mkandes@comet-ln3 gpse]$ ls
CHANGELOG  gpse.input  LICENSE  Makefile  README  source
[mkandes@comet-ln3 gpse]$ cat /etc/os-release | grep PRETTY_NAME
PRETTY_NAME="CentOS Linux 7 (Core)"
[mkandes@comet-ln3 gpse]$ uname -a
Linux comet-ln3.sdsc.edu 3.10.0-957.12.2.el7.x86_64 #1 SMP Tue May 14 21:24:32 U
TC 2019 x86_64 x86_64 x86_64 GNU/Linux
[mkandes@comet-ln3 gpse]$ module list
Currently Loaded Modulefiles:
  1) intel/2018.1.163    2) mvapich2_ib/2.3.2
[mkandes@comet-ln3 gpse]$ module purge
[mkandes@comet-ln3 gpse]$ module load gnu/7.2.0
[mkandes@comet-ln3 gpse]$ module load mvapich2_ib/2.3.2
[mkandes@comet-ln3 gpse]$ module list
Currently Loaded Modulefiles:
  1) gnu/7.2.0           2) mvapich2_ib/2.3.2
[mkandes@comet-ln3 gpse]$ make
mpif90 -Jbuild -fimplicit-none -fmodule-private -ffree-form -ffree-line-length-n
one -std=gnu  -fdefault-real-8 -O2 -mtune=native -c source/math.f90 -o build/mat
h.o
source/math.f90:45:23:

       USE, INTRINSIC :: ISO_FORTRAN_ENV
                         1
Warning: Use of the NUMERIC_STORAGE_SIZE named constant from intrinsic module IS
O_FORTRAN_ENV at (1) is incompatible with option -fdefault-real-8
mpif90 -Jbuild -fimplicit-none -fmodule-private -ffree-form -ffree-line-length-n
one -std=gnu  -fdefault-real-8 -O2 -mtune=native -c source/io.f90 -o build/io.o
source/io.f90:45:23:

       USE, INTRINSIC :: ISO_FORTRAN_ENV
                         1
Warning: Use of the NUMERIC_STORAGE_SIZE named constant from intrinsic module IS
O_FORTRAN_ENV at (1) is incompatible with option -fdefault-real-8
mpif90 -Jbuild -fimplicit-none -fmodule-private -ffree-form -ffree-line-length-n
```

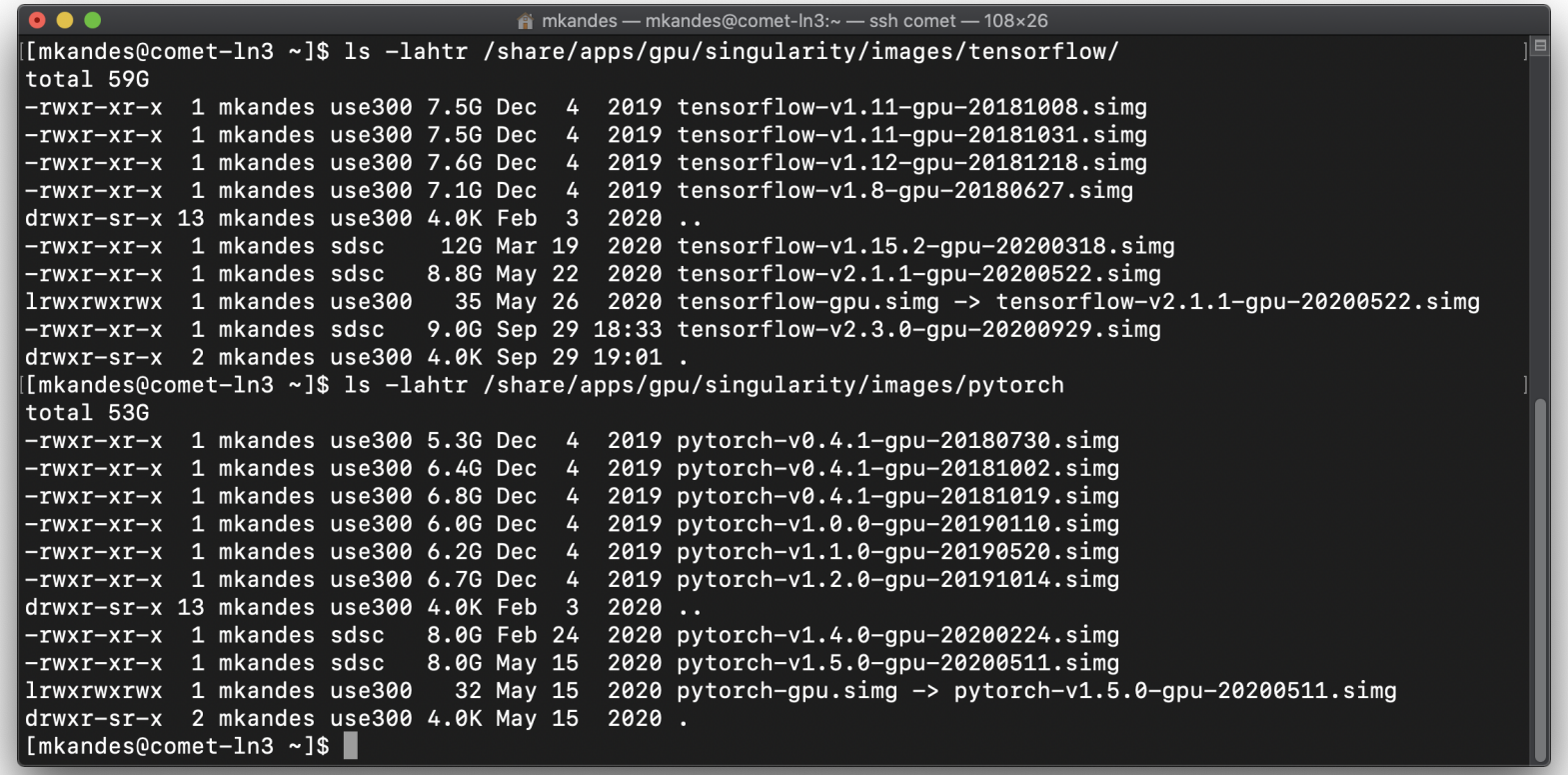
mkandes — mckandes@login2.stampede2:~/gpse — ssh stampede2 — 80×43

```
login2.stampede2(1002)$ git clone https://github.com/mkandes/gpse.git
Cloning into 'gpse'...
remote: Enumerating objects: 528, done.
remote: Total 528 (delta 0), reused 0 (delta 0), pack-reused 528
Receiving objects: 100% (528/528), 311.06 KiB | 2.06 MiB/s, done.
Resolving deltas: 100% (347/347), done.
login2.stampede2(1003)$ cd gpse/
login2.stampede2(1004)$ ls
CHANGELOG  gpse.input  LICENSE  Makefile  README  source
login2.stampede2(1005)$ cat /etc/os-release | grep PRETTY_NAME
PRETTY_NAME="CentOS Linux 7 (Core)"
login2.stampede2(1006)$ uname -a
Linux login2.stampede2.tacc.utexas.edu 3.10.0-957.5.1.el7.x86_64 #1 SMP Fri Feb
1 14:54:57 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
login2.stampede2(1007)$ module list

Currently Loaded Modules:
  1) intel/18.0.2      4) git/2.24.1       7) cmake/3.16.1
  2) libfabric/1.7.0   5) autotools/1.1    8) xalt/2.8
  3) impi/18.0.2       6) python2/2.7.15   9) TACC



login2.stampede2(1008)$ sed -i 's/ := gfortran/ := ifort/g' Makefile
login2.stampede2(1009)$ make
mpif90 -Jbuild -implicitnone -free -stand none -module build  -real-size 64 -ipo
 -O3 -no-prec-div -fp-model fast=2 -xHost -c source/math.f90 -o build/math.o
ifort: command line warning #10006: ignoring unknown option '-Jbuild'
mpif90 -Jbuild -implicitnone -free -stand none -module build  -real-size 64 -ipo
 -O3 -no-prec-div -fp-model fast=2 -xHost -c source/grid.f90 -o build/grid.o
ifort: command line warning #10006: ignoring unknown option '-Jbuild'
source/grid.f90(255): warning #6178: The return value of this FUNCTION has not b
een defined.   [GRID_BINARY_SEARCH]
      INTEGER RECURSIVE FUNCTION grid_binary_search()
--------------------------------^
mpif90 -Jbuild -implicitnone -free -stand none -module build  -real-size 64 -ipo
 -O3 -no-prec-div -fp-model fast=2 -xHost -c source/pmca.f90 -o build/pmca.o
ifort: command line warning #10006: ignoring unknown option '-Jbuild'
mpif90 -Jbuild -implicitnone -free -stand none -module build  -real-size 64 -ipo
 -O3 -no-prec-div -fp-model fast=2 -xHost -c source/vex.f90 -o build/vex.o
ifort: command line warning #10006: ignoring unknown option '-Jbuild'
mpif90 -Jbuild -implicitnone -free -stand none -module build  -real-size 64 -ipo
 -O3 -no-prec-div -fp-model fast=2 -xHost -c source/rot.f90 -o build/rot.o
```

What is a (Software) Container?

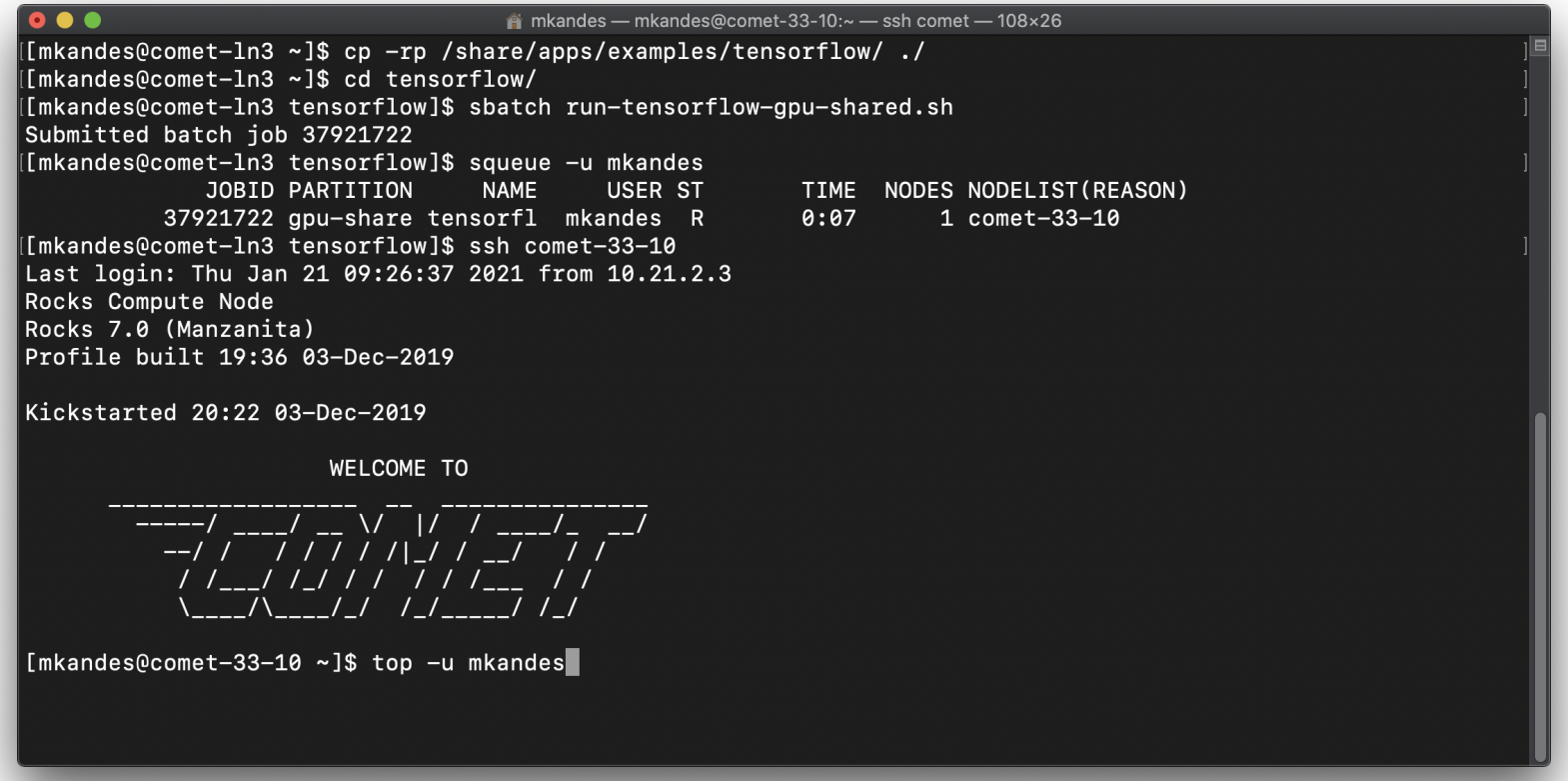
A (software) container is an abstraction for a set of technologies that aim to solve the the problem of how to get software to run reliably when moved from one computing environment to another.

Container Image



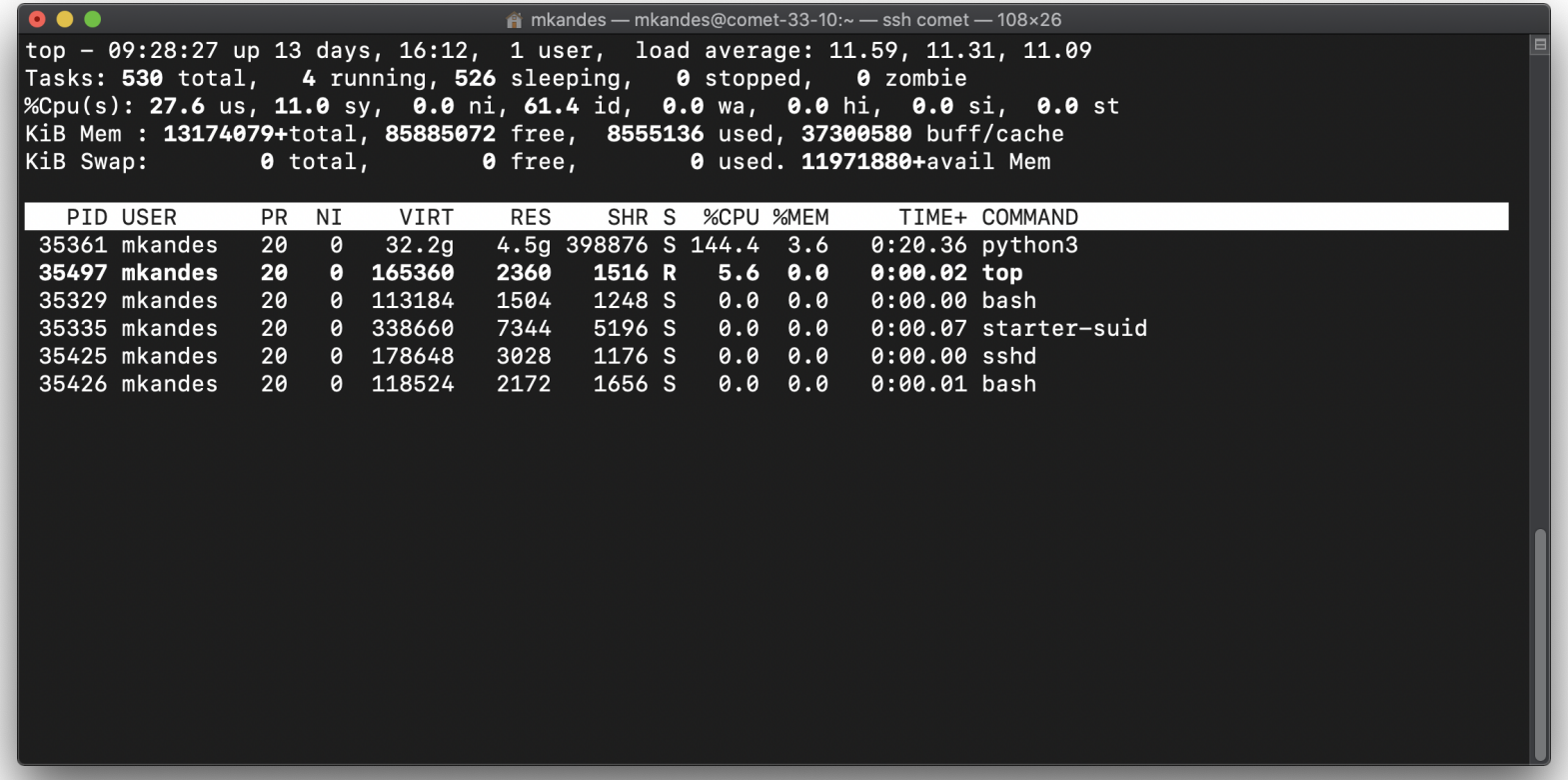
A container image is simply a file (or collection of files) saved on disk that stores everything you need to run a target application or applications: code, runtime, system tools, libraries, etc.

Container Process



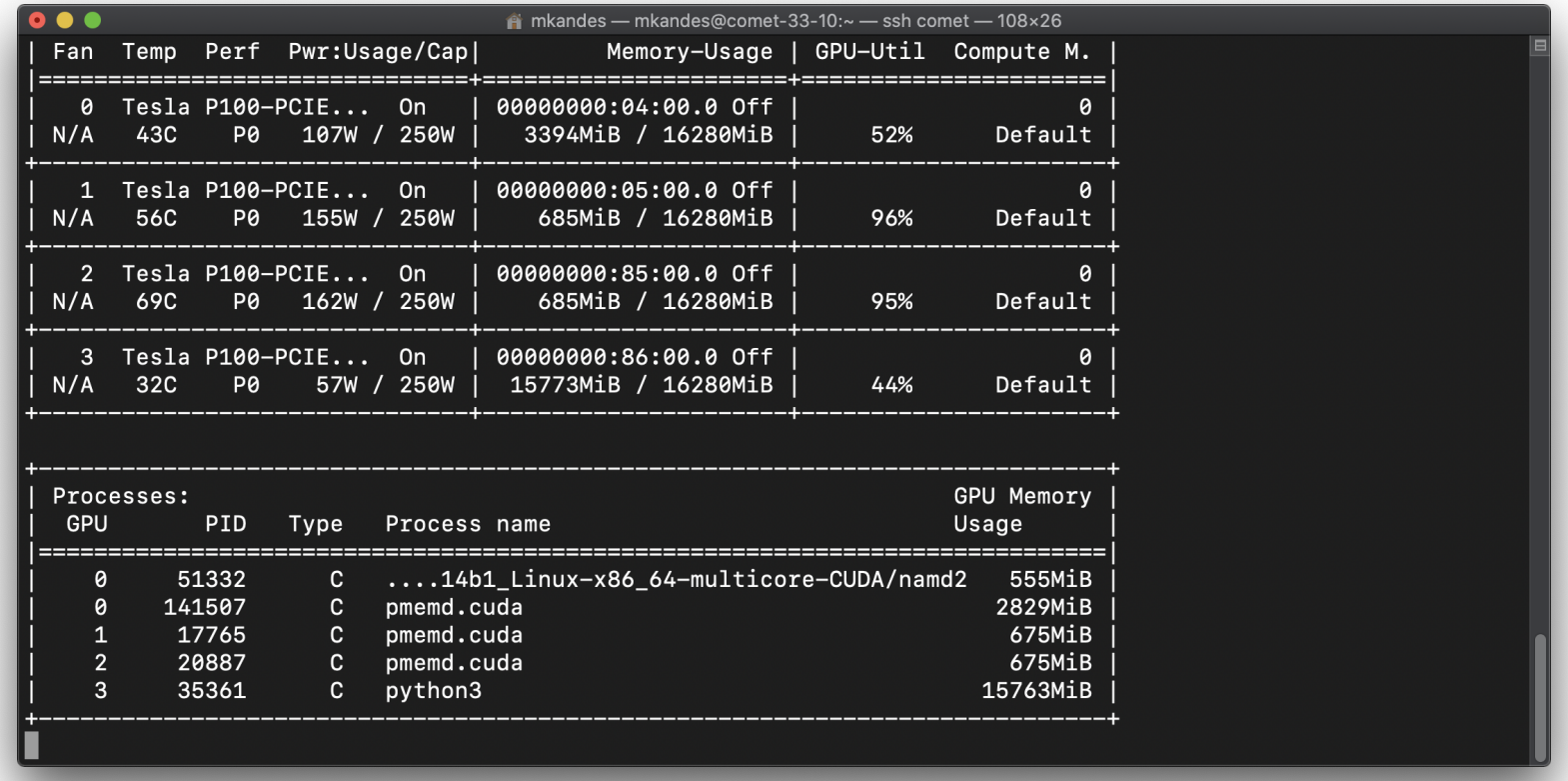
A container process is simply a standard (Linux) process running on top of the underlying host's operating system and kernel, but whose software environment is defined by the contents of the container image.

Container Process



A 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 Process



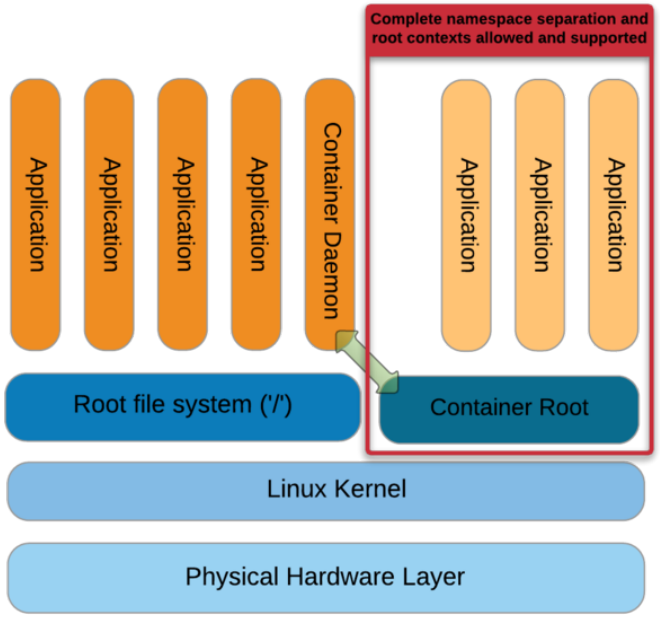
A container process is simply a standard (Linux) process running on top of the underlying host's operating system and kernel, but whose software environment is defined by the contents of the container image.

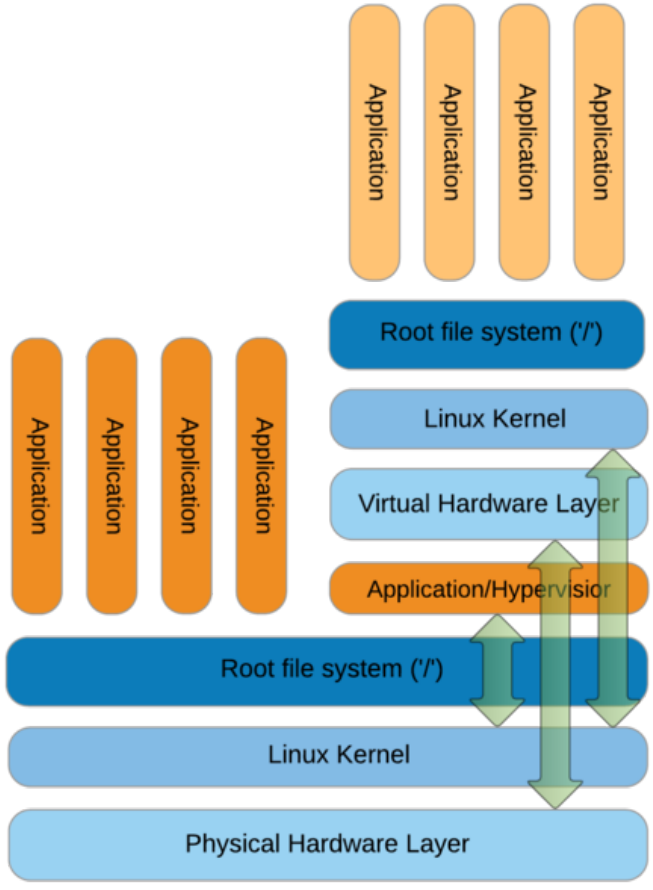
Container : Supercomputer :: Construct : Matrix

“ ... 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 and, thus, are able to achieve similar performance 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

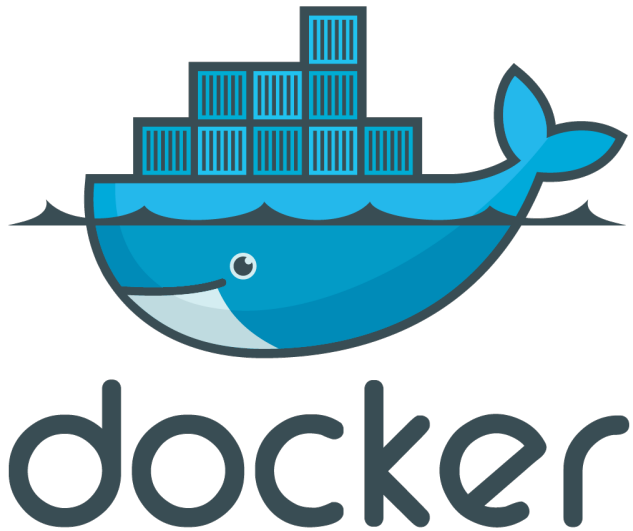
- **Performance**: Near-native application performance
- **Freedom**: Bring your own software environment
- **Reproducibility**: Package complex software applications into easy to manage, verifiable software units
- **Compatibility**: Built on open standards available in all major Linux distributions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

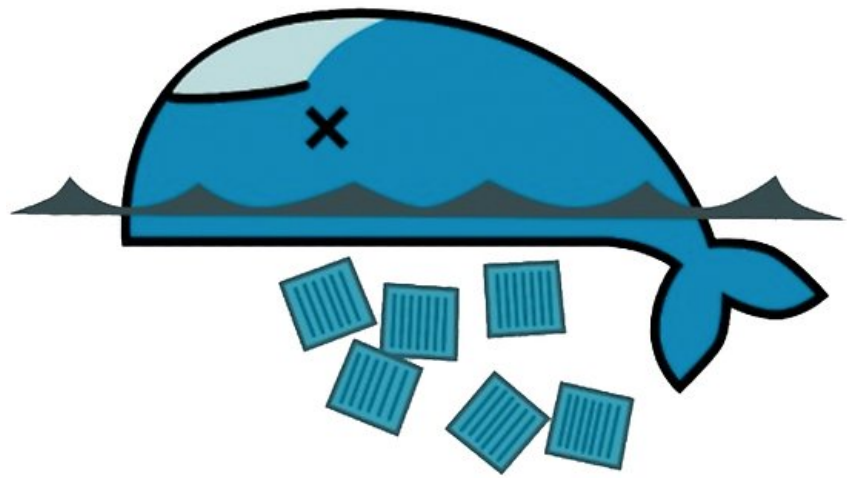
- Most common container platform in use today
- Provides tools and utilities to create, maintain, distribute, and run containers images
- Designed to accommodate network-centric services (web servers, databases, etc)
- Easy to install, well-documentated, and large, well-developed user community and container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



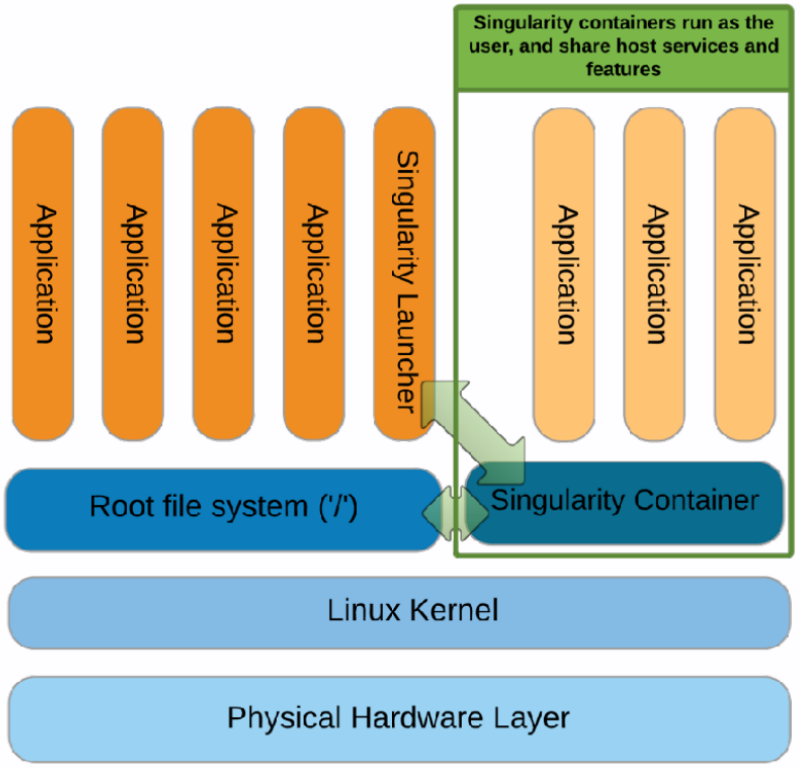
Singularity: A Container Platform for HPC

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

`https://github.com/hpcng/singularity`
`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

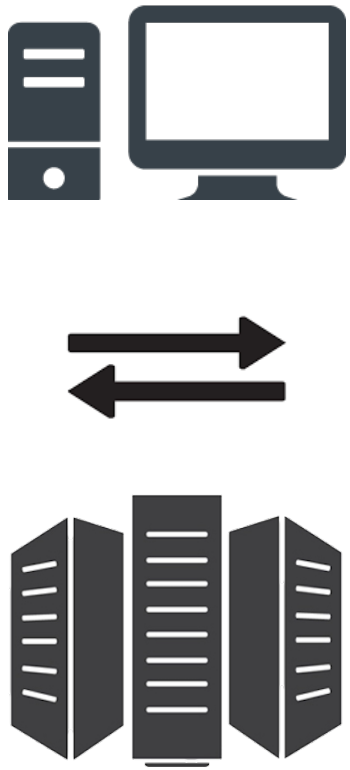


Most Common Singularity Use Cases

- 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:
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:

* 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.

Essential Singularity

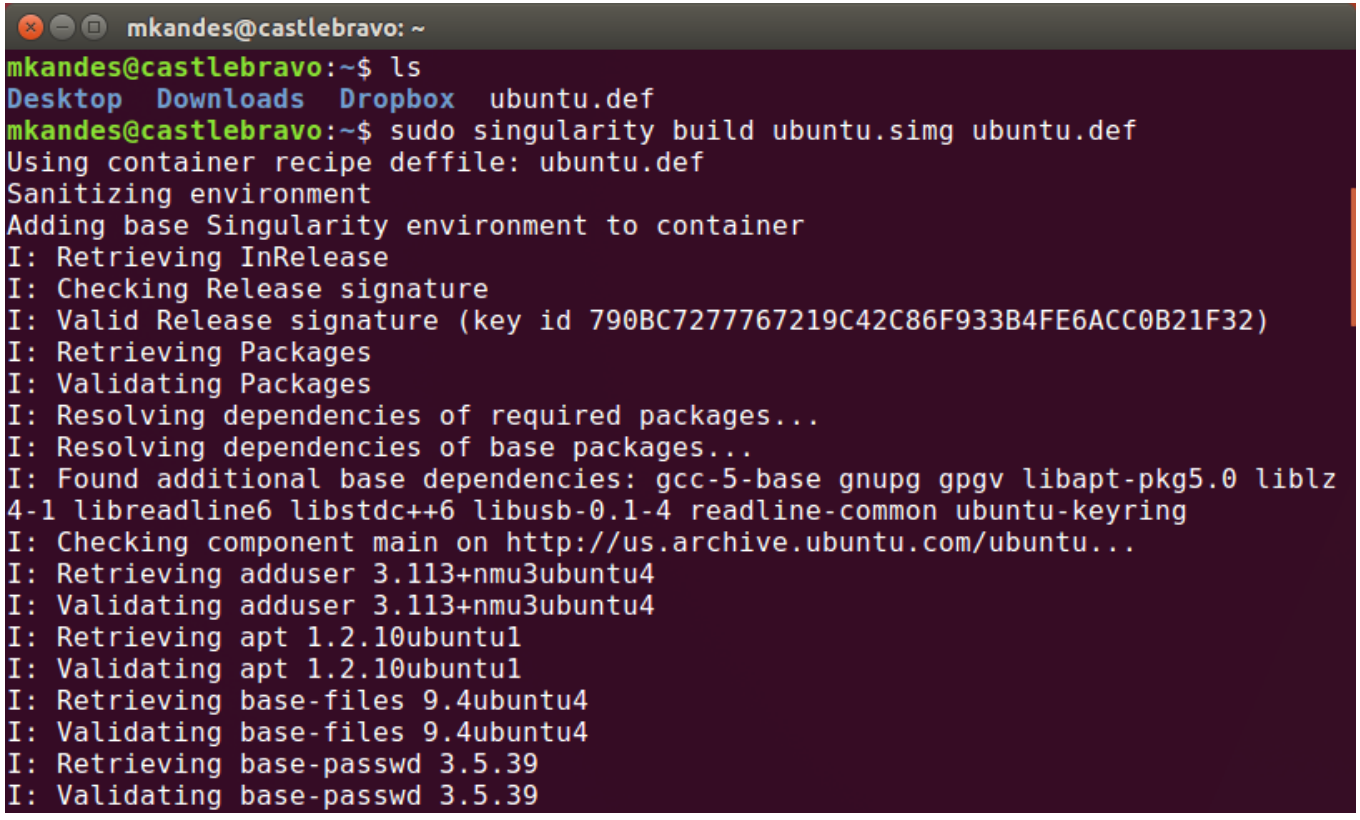
The main Singularity command

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

has three essential subcommands:

- `build`: Build your own container from scratch using a Singularity definition 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.

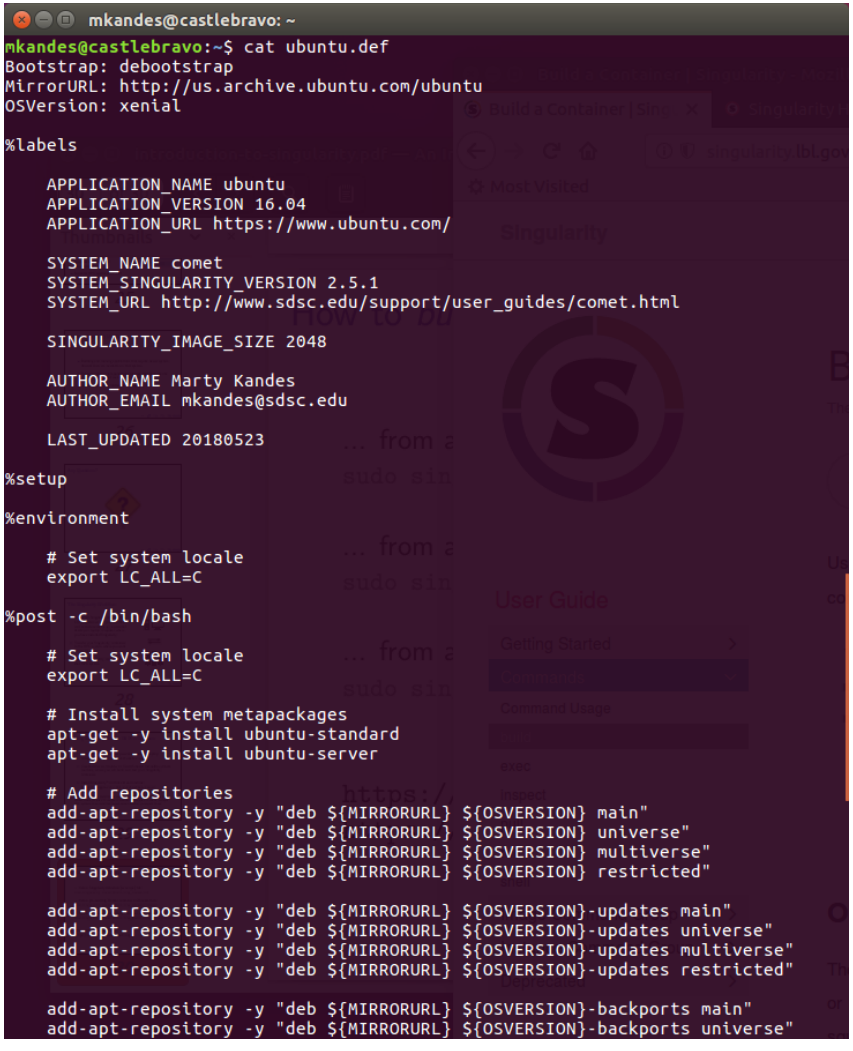
Building a Singularity Container




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

Singularity Definition File

- A Singularity definition 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.



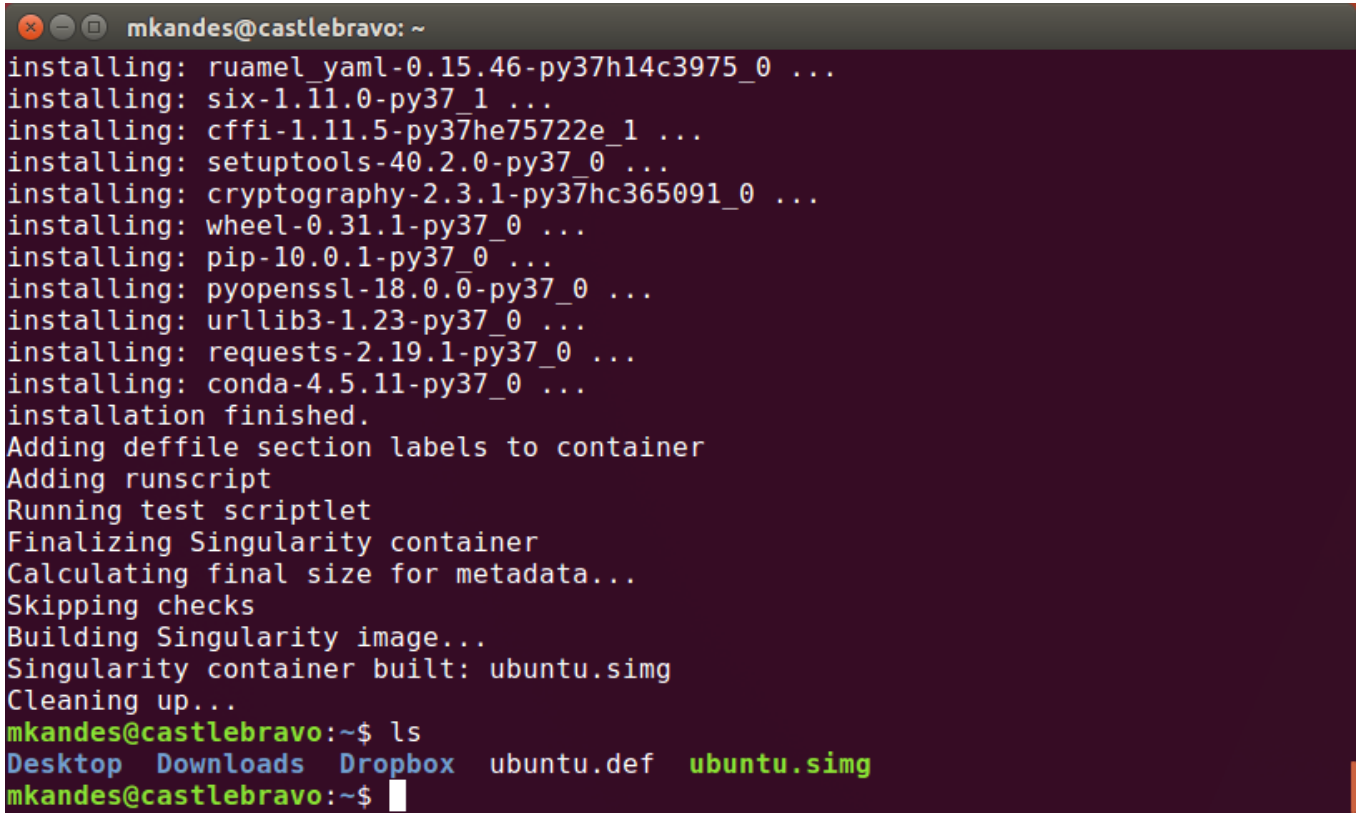
naked-singularity

- A repository of definition 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`

Building a Singularity Container

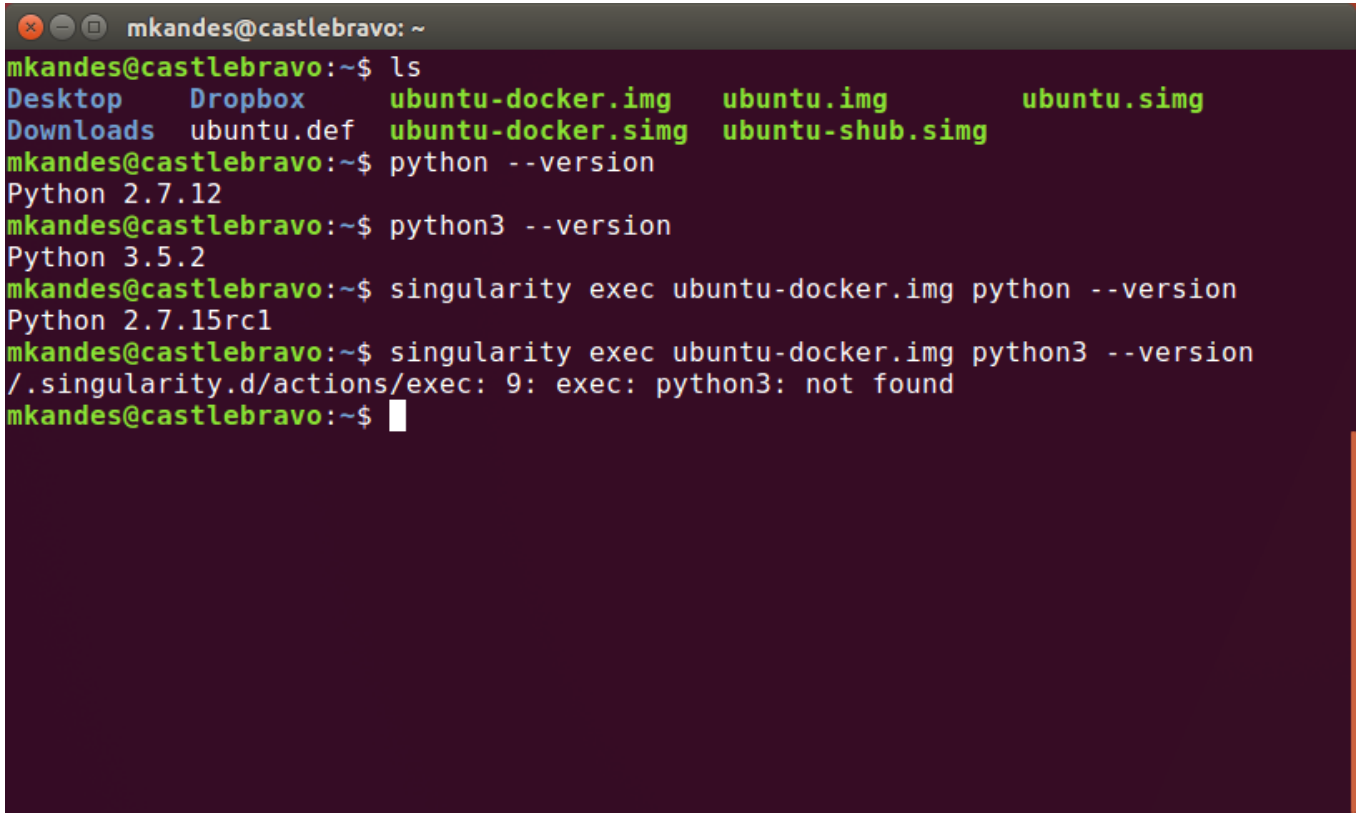



```
sudo singularity build ubuntu.sif ubuntu.def
```

Interacting with a Singularity Container


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

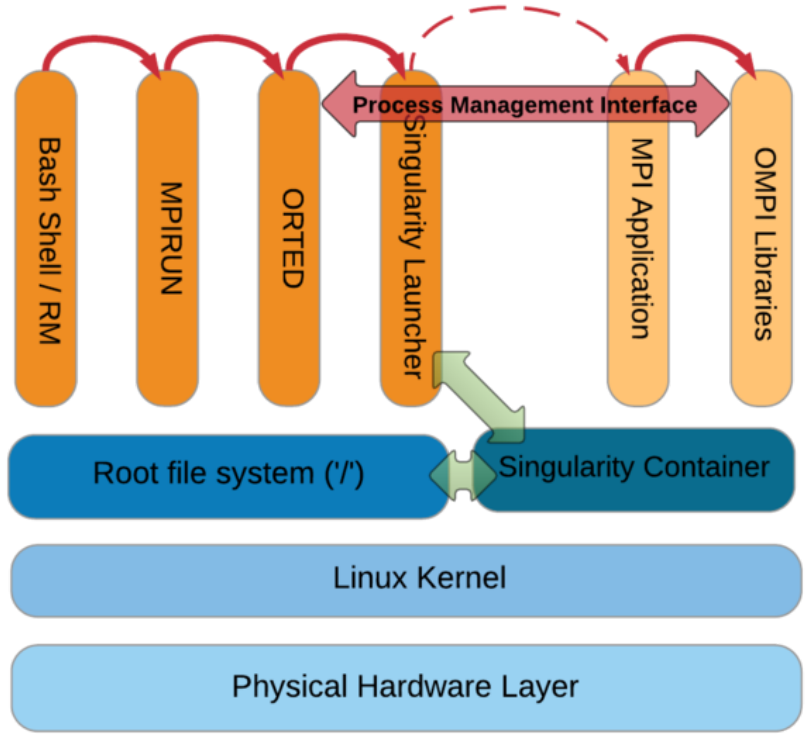
Running a Singularity Container




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

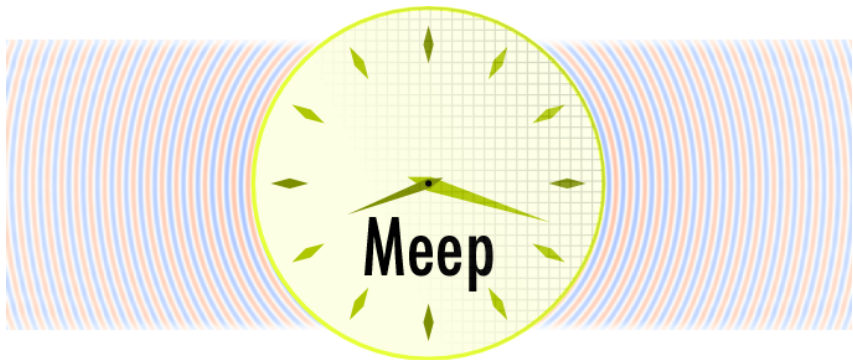
MPI-based Singularity Containers

- 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 most native software environments.



MPI-based Singularity Containers: MEEP

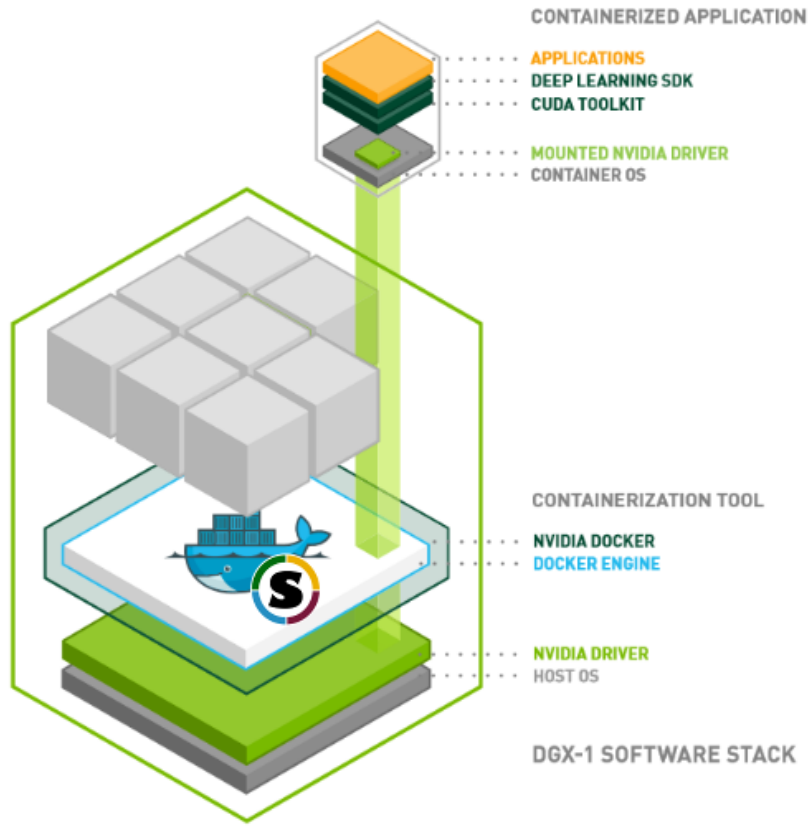
mkandes@comet-ln2:~/Software/meep

```
#!/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
```

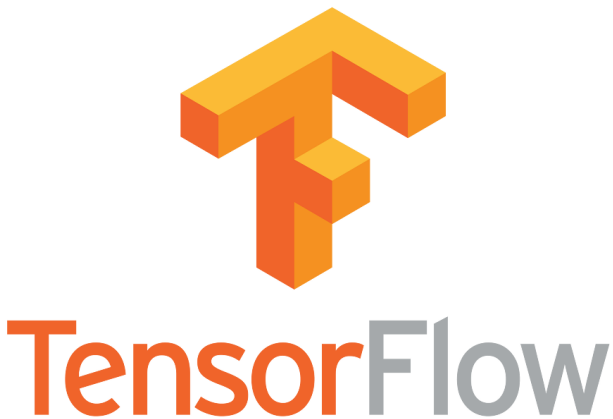
GPU-accelerated Singularity Containers

- 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, making it difficult to install on systems with older operating systems.



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 tensorflow-gpu.sif python mnist_deep.py
```

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.

Additional References

- Singularity User Guide:
  `https://sylabs.io/guides/latest/user-guide/`
- Example Singularity Definition Files:
  `https://github.com/mkandes/naked-singularity`
- Talks @ 1st Singularity User Group Meeting:
  `https://sylabs.io/videos`