

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)

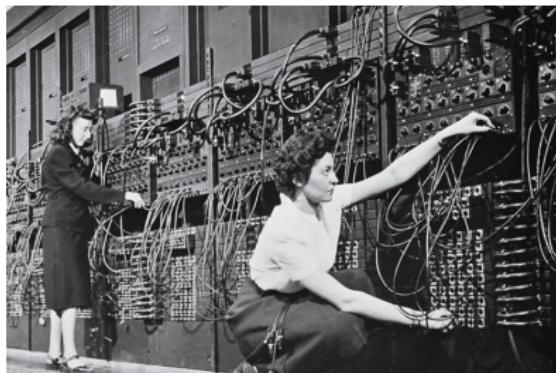
- ▶ Standardized - containers of known dimensions and permissible weight tolerances
- ▶ Efficient - portable containers allow fast loading and unloading from multiple modes of transportation
- ▶ Secure - goods remained stored within the same container during transport
- ▶ Global - cost effective to ship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



Computing ≈ Shipping?



Evolution of Software Deployment



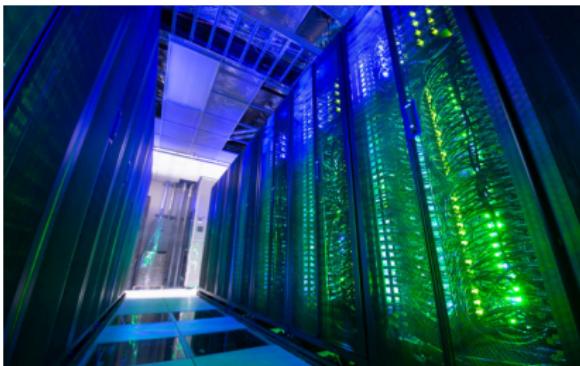
```
SONS> aws ec2 help
EC2()

NAME
    ec2 - 

DESCRIPTION
    Amazon Elastic Compute Cloud (Amazon EC2) provides resizable computing
    capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates
    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



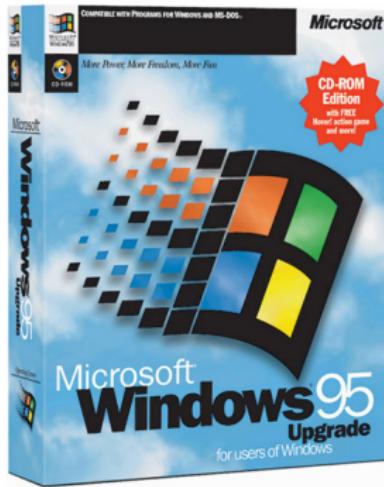
Comet vs. Stampede2

```
mkanedes@comet-ln3:~$ git clone https://github.com/mkanedes/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.
[mkanedes@comet-ln3 ~]$ cd gpse/
[mkanedes@comet-ln3 gpse]$ ls
CHANGELOG gpse.input LICENSE Makefile README source
[mkanedes@comet-ln3 gpse]$ cat /etc/os-release | grep PRETTY_NAME
PRETTY_NAME="CentOS Linux 7 (Core)"
[mkanedes@comet-ln3 gpse]$ uname -a
Linux comet-ln3.sdsu.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
[mkanedes@comet-ln3 gpse]$ module list
Currently Loaded Modulefiles:
 1) intel/2018.1.163  2) mvapich2_ib/2.3.2
[mkanedes@comet-ln3 gpse]$ module purge
[mkanedes@comet-ln3 gpse]$ module load gnu/7.2.0
[mkanedes@comet-ln3 gpse]$ module load mvapich2_ib/2.3.2
[mkanedes@comet-ln3 gpse]$ module list
Currently Loaded Modulefiles:
 1) gnu/7.2.0          2) mvapich2_ib/2.3.2
[mkanedes@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/math.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
```

```
mkanedes@stampede2:~$ git clone https://github.com/mkanedes/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.
[mkanedes@stampede2:~]$ cd gpse/
[mkanedes@stampede2:~]$ ls
CHANGELOG gpse.input LICENSE Makefile README source
[mkanedes@stampede2:~]$ cat /etc/os-release | grep PRETTY_NAME
PRETTY_NAME="CentOS Linux 7 (Core)"
[mkanedes@stampede2:~]$ 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
[mkanedes@stampede2:~]$ 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

[mkanedes@stampede2:~]$ sed -i 's/ := gfortran/ := ifort/g' Makefile
[mkanedes@stampede2:~]$ make
mpiF90 -Jbuild -fimplicitnone -free -stand none -module build -real-size 64 -ipo
-O3 -no-prec-div -fp-model fast2 -xhost -c source/math.f90 -o build/math.o
ifort: command line warning #10006: ignoring unknown option '-Jbuild'
mpiF90 -Jbuild -fimplicitnone -free -stand none -module build -real-size 64 -ipo
-O3 -no-prec-div -fp-model fast2 -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 -fimplicitnone -free -stand none -module build -real-size 64 -ipo
-O3 -no-prec-div -fp-model fast2 -xhost -c source/pmc.a.f90 -o build/pmc.a.o
ifort: command line warning #10006: ignoring unknown option '-Jbuild'
mpiF90 -Jbuild -fimplicitnone -free -stand none -module build -real-size 64 -ipo
-O3 -no-prec-div -fp-model fast2 -xhost -c source/vex.f90 -o build/vex.o
ifort: command line warning #10006: ignoring unknown option '-Jbuild'
mpiF90 -Jbuild -fimplicitnone -free -stand none -module build -real-size 64 -ipo
-O3 -no-prec-div -fp-model fast2 -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 problem of how to get software to run reliably when moved from one computing environment to another.

Container Image

```
[mkandes@comet-ln3 ~]$ ls -lahtr /share/apps/gpu/singularity/images/tensorflow/
total 59G
-rwxr-xr-x 1 mkandes use300 7.5G Dec  4 2019 tensorflow-v1.11-gpu-20181008.simg
-rwxr-xr-x 1 mkandes use300 7.5G Dec  4 2019 tensorflow-v1.11-gpu-20181031.simg
-rwxr-xr-x 1 mkandes use300 7.6G Dec  4 2019 tensorflow-v1.12-gpu-20181218.simg
-rwxr-xr-x 1 mkandes use300 7.1G Dec  4 2019 tensorflow-v1.8-gpu-20180627.simg
drwxr-sr-x 13 mkandes use300 4.0K Feb  3 2020 ..
-rwxr-xr-x 1 mkandes sdsc  12G Mar 19 2020 tensorflow-v1.15.2-gpu-20200318.simg
-rwxr-xr-x 1 mkandes sdsc  8.8G May 22 2020 tensorflow-v2.1.1-gpu-20200522.simg
lrwxrwxrwx 1 mkandes use300  35 May 26 2020 tensorflow-gpu.simg -> tensorflow-v2.1.1-gpu-20200522.simg
-rwxr-xr-x 1 mkandes sdsc  9.0G Sep 29 18:33 tensorflow-v2.3.0-gpu-20200929.simg
drwxr-sr-x 2 mkandes use300 4.0K Sep 29 19:01 .
[lmkandes@comet-ln3 ~]$ ls -lahtr /share/apps/gpu/singularity/images/pytorch
total 53G
-rwxr-xr-x 1 mkandes use300 5.3G Dec  4 2019 pytorch-v0.4.1-gpu-20180730.simg
-rwxr-xr-x 1 mkandes use300 6.4G Dec  4 2019 pytorch-v0.4.1-gpu-20181002.simg
-rwxr-xr-x 1 mkandes use300 6.8G Dec  4 2019 pytorch-v0.4.1-gpu-20181019.simg
-rwxr-xr-x 1 mkandes use300 6.0G Dec  4 2019 pytorch-v1.0.0-gpu-20190110.simg
-rwxr-xr-x 1 mkandes use300 6.26G Dec  4 2019 pytorch-v1.0-gpu-20190520.simg
-rwxr-xr-x 1 mkandes use300 6.7G Dec  4 2019 pytorch-v1.1.0-gpu-20191014.simg
drwxr-sr-x 13 mkandes use300 4.0K Feb  3 2020 ..
-rwxr-xr-x 1 mkandes sdsc  8.0G Feb 24 2020 pytorch-v1.4.0-gpu-20200224.simg
-rwxr-xr-x 1 mkandes sdsc  8.0G May 15 2020 pytorch-v1.5.0-gpu-20200511.simg
lrwxrwxrwx 1 mkandes use300  32 May 15 2020 pytorch-gpu.simg -> pytorch-v1.5.0-gpu-20200511.simg
drwxr-sr-x 2 mkandes use300 4.0K May 15 2020 .
[lmkandes@comet-ln3 ~]$
```

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

```
top - 09:28:27 up 13 days, 16:12,  1 user,  load average: 11.59, 11.31, 11.09
Tasks: 530 total,   4 running, 526 sleeping,   0 stopped,   0 zombie
%Cpu(s): 27.6 us, 11.0 sy,  0.0 ni, 61.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 13174079+total, 85885072 free, 8555136 used, 37300580 buff/cache
KiB Swap:     0 total,      0 free,      0 used. 11971880+avail Mem

PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
35361 mkandes  20   0  32.2g  4.5g 398876 S 144.4  3.6  0:20.36 python3
35497 mkandes  20   0 165360  2360 1516 R  5.6  0.0  0:00.02 top
35329 mkandes  20   0 113184  1504 1248 S  0.0  0.0  0:00.00 bash
35335 mkandes  20   0 338660  7344 5196 S  0.0  0.0  0:00.07 starter-suid
35425 mkandes  20   0 178648  3028 1176 S  0.0  0.0  0:00.00 sshd
35426 mkandes  20   0 118524  2172 1656 S  0.0  0.0  0:00.01 bash
```

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

```
mkanedes@mkandes-comet-33-10:~$ ssh comet - 108x26
| Fan Temp Perf Pwr:Usage/Cap|      Memory-Usage | GPU-Util Compute M. |
+=====+=====+=====+=====+
|   0 Tesla P100-PCIE... On | 00000000:04:00.0 Off |          0 |
| N/A 43C     P0 107W / 250W | 3394MiB / 16280MiB | 52% Default |
+-----+
|   1 Tesla P100-PCIE... On | 00000000:05:00.0 Off |          0 |
| N/A 56C     P0 155W / 250W | 685MiB / 16280MiB | 96% Default |
+-----+
|   2 Tesla P100-PCIE... On | 00000000:85:00.0 Off |          0 |
| N/A 69C     P0 162W / 250W | 685MiB / 16280MiB | 95% Default |
+-----+
|   3 Tesla P100-PCIE... On | 00000000:86:00.0 Off |          0 |
| N/A 32C     P0 57W / 250W | 15773MiB / 16280MiB | 44% Default |
+-----+
+-----+
| Processes:                               GPU Memory |
| GPU     PID  Type  Process name           Usage    |
+=====+=====+=====+=====
|   0     51332  C  ....14b1_Linux-x86_64-multicore-CUDA/namd2 555MiB |
|   0     141507  C  pmemd.cuda               2829MiB |
|   1     17765   C  pmemd.cuda               675MiB  |
|   2     20887   C  pmemd.cuda               675MiB  |
|   3     35361   C  python3                  15763MiB |
+-----+
```

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.

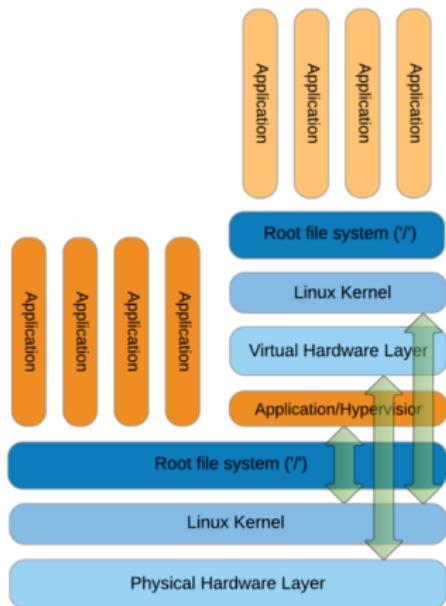
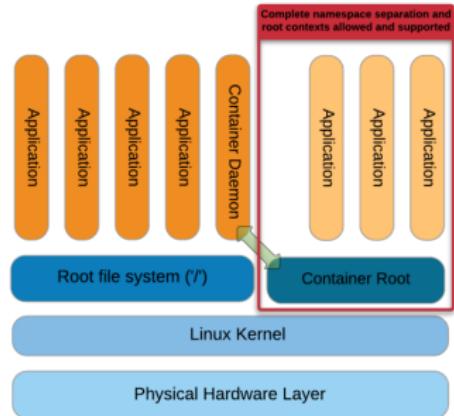
Container : Supercomputer :: Construct : Matrix

“ ... 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

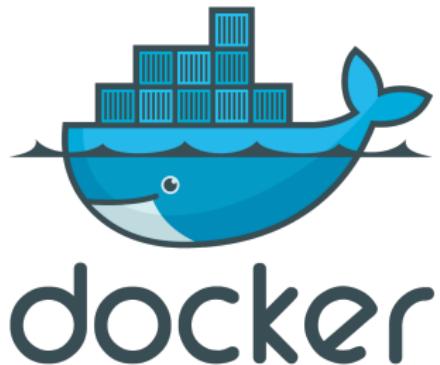
- ▶ **Performance:** Near-native application performance
- ▶ **Freedom:** Bring your own software environment
- ▶ **Reproducibility:** Package complex software applications into easy to manage, verifiable software units
- ▶ **Compatibility:** Built on open standards available in all major Linux distributions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

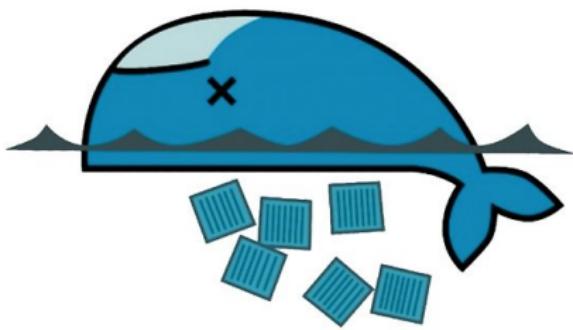
- ▶ Most common container platform in use today
- ▶ Provides tools and utilities to create, maintain, distribute, and run containers images
- ▶ Designed to accommodate network-centric services (web servers, databases, etc)
- ▶ Easy to install, well-documented, and large, well-developed user community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



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

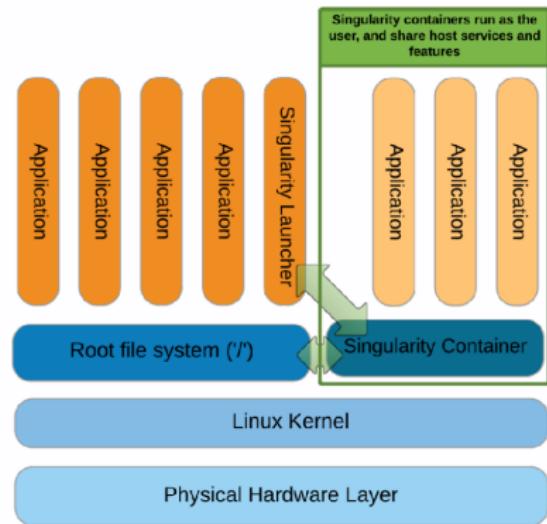


[https://github.com/hpcng/
singularity](https://github.com/hpcng/singularity)

<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.

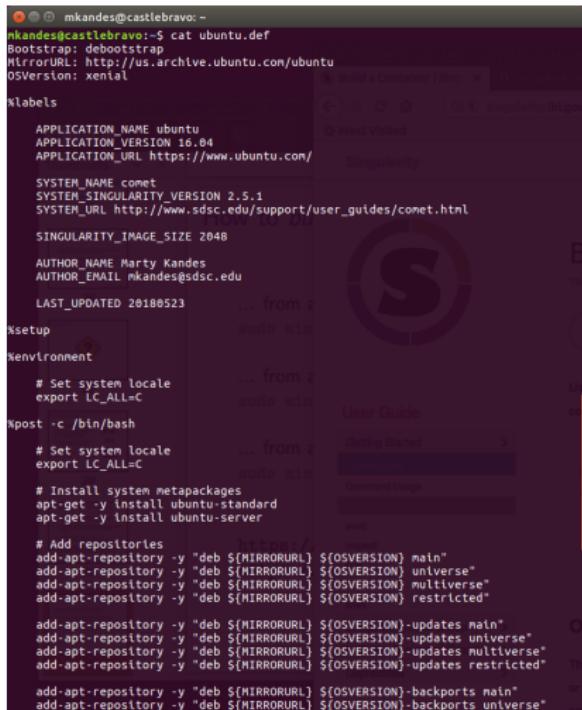
Building a Singularity Container

```
mkanedes@castlebravo:~  
mkanedes@castlebravo:~$ ls  
Desktop Downloads Dropbox ubuntu.def  
mkanedes@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 liblz  
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 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.



```
mikandes@castlebravo:~$ cat ubuntu.def
Bootstrap: debootstrap
MirrorURL: http://us.archive.ubuntu.com/ubuntu
OSVersion: xenial

xlabels
    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 2048
    AUTHOR_NAME Marty Kandes
    AUTHOR_EMAIL mikandes@sdsc.edu
    LAST_UPDATED 20180523

ksetup
environment
    # Set system locale
    export LC_ALL=C
    post -c /bin/bash
        # Set system locale
        export LC_ALL=C
        # Install system metapackages
        apt-get -y install ubuntu-standard
        apt-get -y install ubuntu-server
        # Add repositories
        add-apt-repository -y "deb ${MIRRORURL} ${OSVERSION} main"
        add-apt-repository -y "deb ${MIRRORURL} ${OSVERSION} universe"
        add-apt-repository -y "deb ${MIRRORURL} ${OSVERSION} multiverse"
        add-apt-repository -y "deb ${MIRRORURL} ${OSVERSION} restricted"
        add-apt-repository -y "deb ${MIRRORURL} ${OSVERSION}-updates main"
        add-apt-repository -y "deb ${MIRRORURL} ${OSVERSION}-updates universe"
        add-apt-repository -y "deb ${MIRRORURL} ${OSVERSION}-updates multiverse"
        add-apt-repository -y "deb ${MIRRORURL} ${OSVERSION}-updates restricted"
        add-apt-repository -y "deb ${MIRRORURL} ${OSVERSION}-backports main"
        add-apt-repository -y "deb ${MIRRORURL} ${OSVERSION}-backports universe"
```

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

```
mkandes@castlebravo: ~
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...
mkandes@castlebravo:~$ ls
Desktop  Downloads  Dropbox  ubuntu.def  ubuntu.simg
mkandes@castlebravo:~$
```

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

Interacting with a Singularity Container

```
mkanedes@castlebravo:~$ ls
Desktop  Dropbox  ubuntu-docker.simg  ubuntu.simg
Downloads  ubuntu.def  ubuntu-shub.simg
mkanedes@castlebravo:~$ which python
/usr/bin/python
mkanedes@castlebravo:~$ python --version
Python 2.7.12
mkanedes@castlebravo:~$ which python3
/usr/bin/python3
mkanedes@castlebravo:~$ python3 --version
Python 3.5.2
mkanedes@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
mkanedes@castlebravo:~$ █
```

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

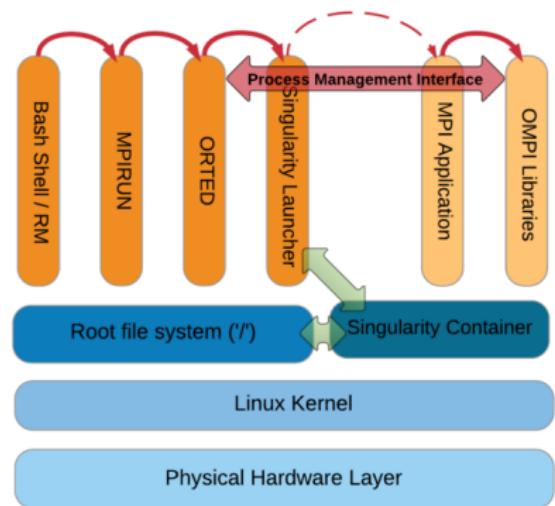
Running a Singularity Container

```
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
```

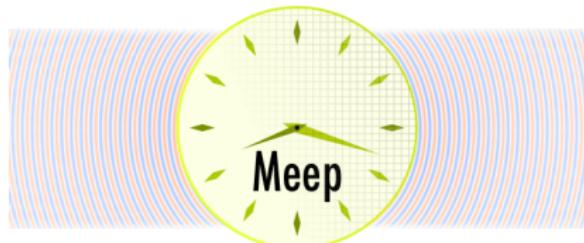
MPI-based Singularity Containers

- ▶ 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'

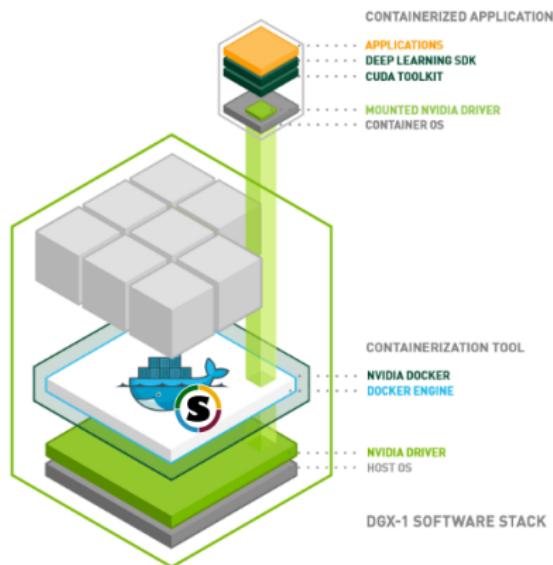
module purge
module load "${COMPILER_MODULE}"
module load "${MPI_MODULE}"
module load "${SINGULARITY_MODULE}"
module list
printenv

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>