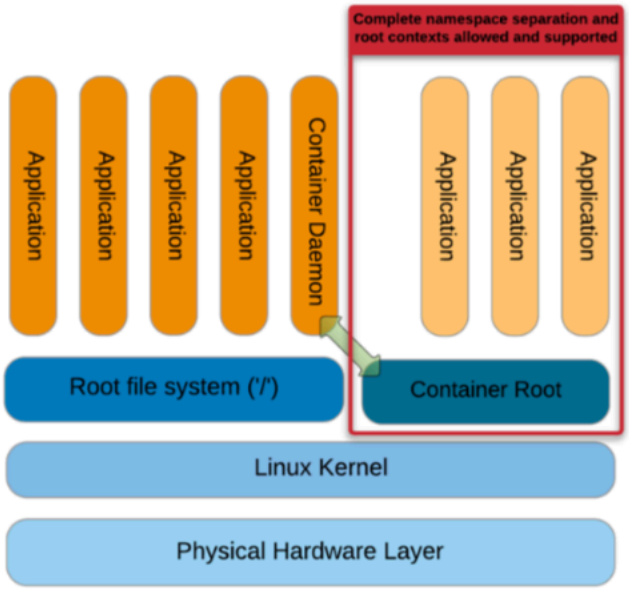
Containers for Scientific and High Performance Computing (HPC)

Mahidhar Tatineni
UCLA
Nov 8, 2019

What is a Container?

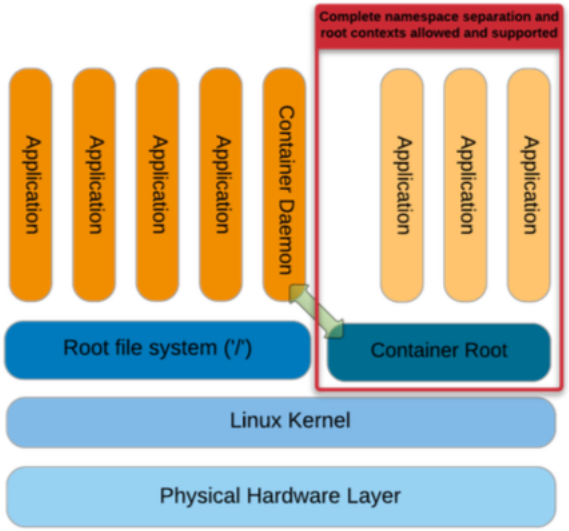
- *At rest*, a container or **container image** is simply a file (or collection of files) saved on disk that stores everything you need to run a target application or applications: code, runtime, system tools, libraries and settings, etc.

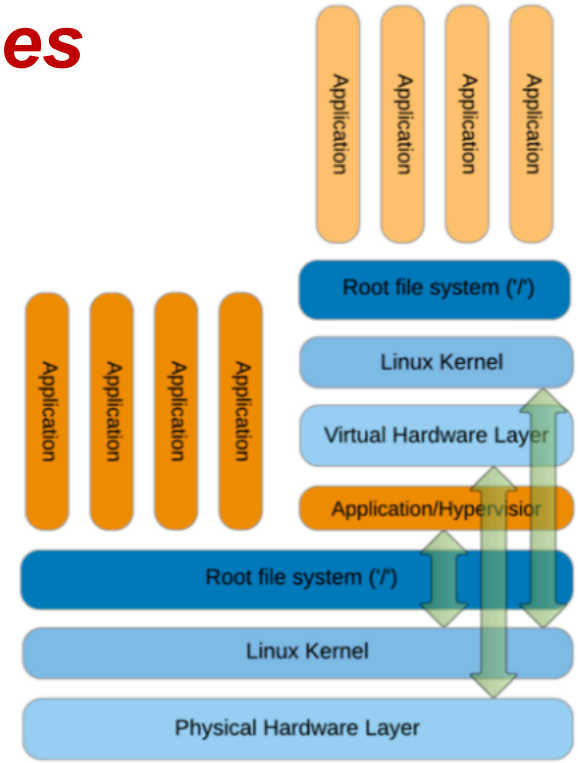
- *In motion*, a container or **container process** is simply a standard (Linux) process running on top of the underlying host's operating system and kernel, but whose software environment is defined by the contents of the container image.



Reference: An Introduction to Singularity: Containers for Scientific and High-Performance Computing
Marty Kandes, SDSC Summer Institute 2019

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 can create a significant performance overhead.

Reference: An Introduction to Singularity: Containers for Scientific and High-Performance Computing
Marty Kandes, SDSC Summer Institute 2019

Containers: Advantages and Limitations

- **Advantages:**
  - *Near native application* ***performance***
  - **Flexible** (bring your own) software environment
  - **Reproducibility**
  - **Compatibility** with most major Linux distributions
  - **Portable** (with some limits detailed below).

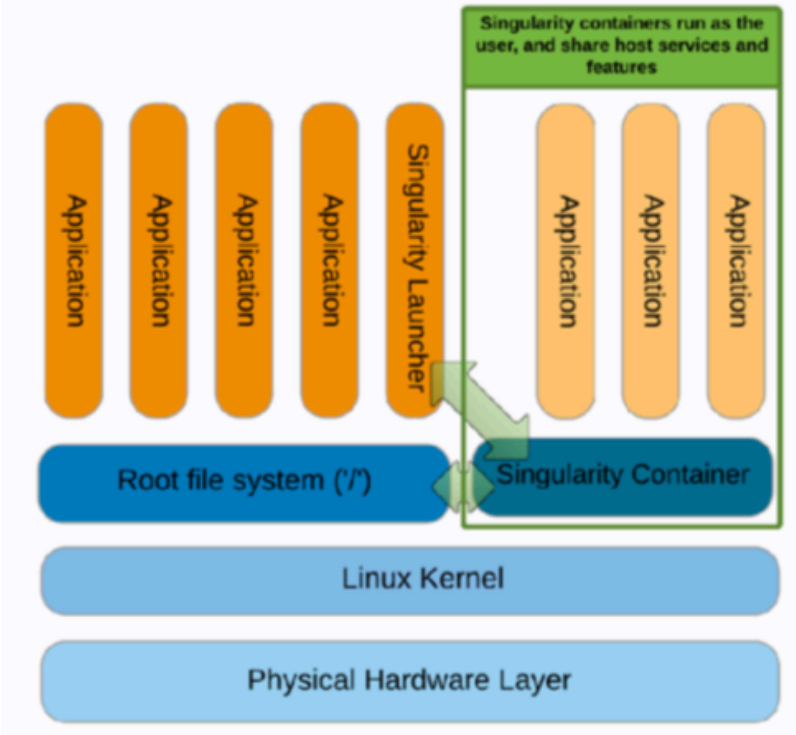
- **Limitations:**
  - **Architecture-dependent (CPU, binary format)**
  - **Portability constraints: kernel, glibc, drivers**

Singularity: Provides Flexibility for OS Environment

- Singularity (http://singularity.lbl.gov) is the container solution on Comet.
- Singularity allows groups to easily migrate complex software stacks from their campus to Comet.
- Singularity runs in user space, and requires very little special support – in fact it actually reduces it in some cases.
- We have roughly 15 groups running this on Comet.
- Applications include: Tensorflow, Torch, Fenics, and custom user applications.
- Docker images can be imported into Singularity.

Features of Singularity

- Each container is a single image file.
- No root owned daemon processes.
- No root escalation, user inside container is the same as the user who started the container.
- Can run in shared/mult—tenant resource environments
- Support for HPC networks (InfiniBand), GPUs
- Support for MPI



Use Case: PDB REDO project

- **Assisted PDB-REDO team in running the complete PDB-REDO pipeline with with 101,570 entries using Gordon and Comet supercomputers**
- **Complex software stack with 50+ independent libraries. Docker container imported via Singularity.**
- **Skill in working at-scale - scheduled computations for 100K+ entries with multiple steps using pylauncher**
- **Singularity gave flexibility to move the complex stack between Gordon and Comet (for large memory) and bind mount scratch directories (different on two machines).**
- **Published paper:**
  - van Beusekom B, Touw WG, Tatineni M, Somani S, Rajagopal G, Luo J, Gilliland GL, Perrakis A, Joosten RP Homology-based hydrogen bond information improves crystallographic structures in the PDB. Protein Science. 2018;27:798-808.

The Singularity Workflow

- **Build** your Singularity containers on a local system where you have root/sudo access. For example: virtual box on your laptop/desktop.
- **Transfer** your container to the HPC system.
- **Run** your Singularity container on the HPC system.

Reference: An Introduction to Singularity: Containers for Scientific and High-Performance Computing
Marty Kandes, SDSC Summer Institute 2019

Essential Singularity Command/Options

- **The main Singularity command**

  singularity [options] **<subcommand>** [subcommand options]

  has three essential subcommands:

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

Singularity Image Sources

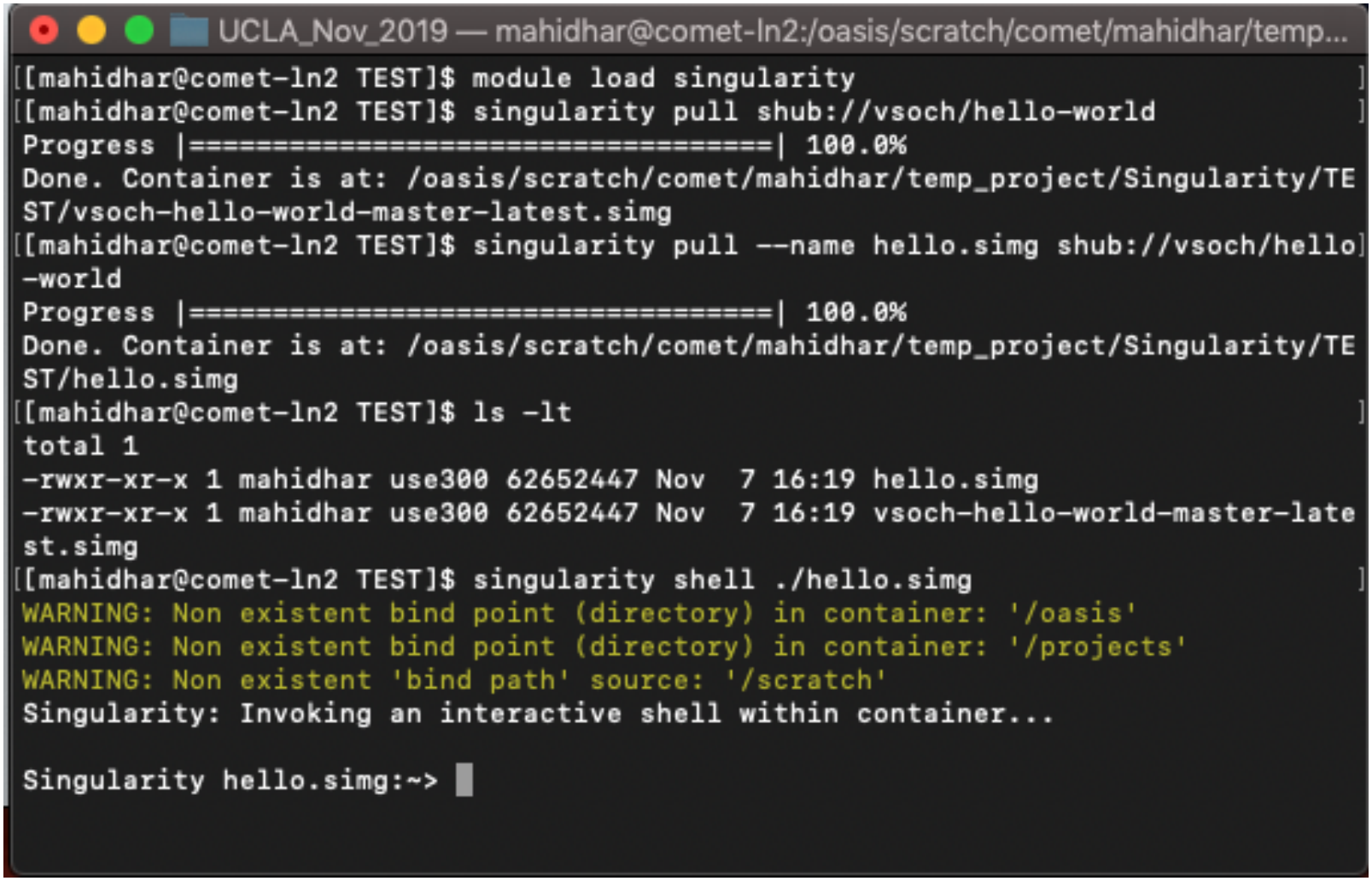
- **SDSC staff have some useful images in:**
  - /share/apps/compute/singularity
  - /share/apps/gpu/singularity
- **Users can build their own images on their laptops/desktops/cloud - as long as you have singularity installed and have root access on your own machine (or VM or cloud instance)**
- **Pull an image from Singularity Hub**
- **Import a docker image**
- **Comet specific documentation available at:**
  - http://www.sdsc.edu/support/user_guides/tutorials/about_comet_singularity_containers.html

Downloading prebuilt images

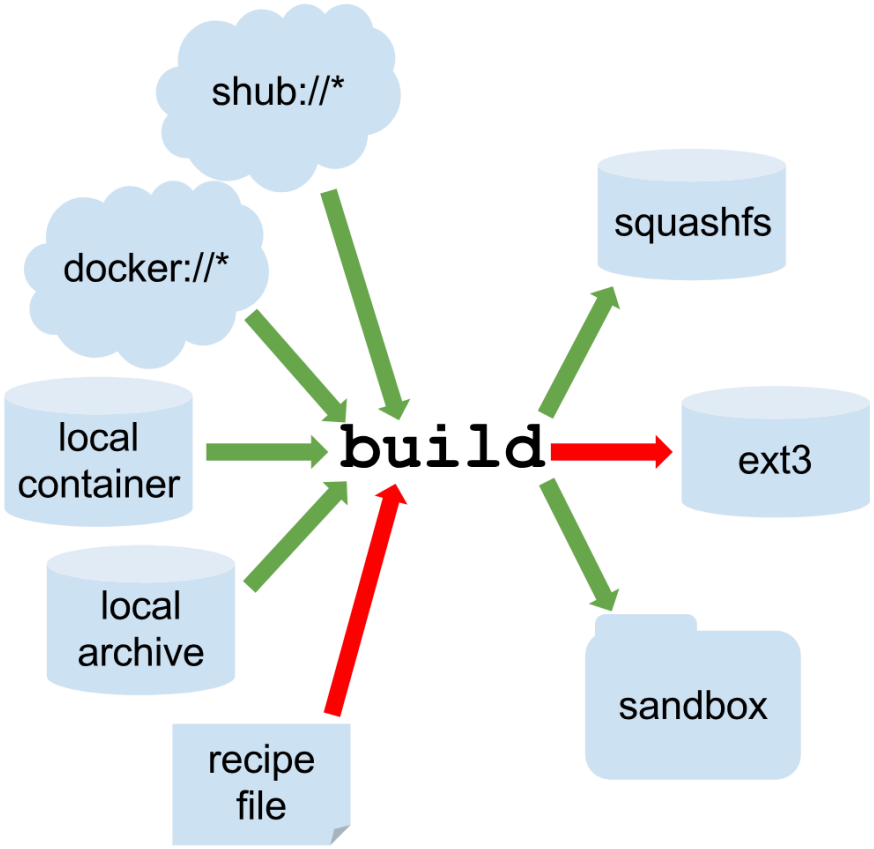
- **singularity pull command can be used to download images from**
  - *Singularity Hub*: e.g. singularity pull shub://vsoch/hello-world
  - *Docker Hub :* e.g. singularity pull --name funny.simg docker://godlovedc/lolcow

- **The pull option simply downloads the image to your system. You can download to a custom name.**

Singularity Pull



Singularity ”build” option



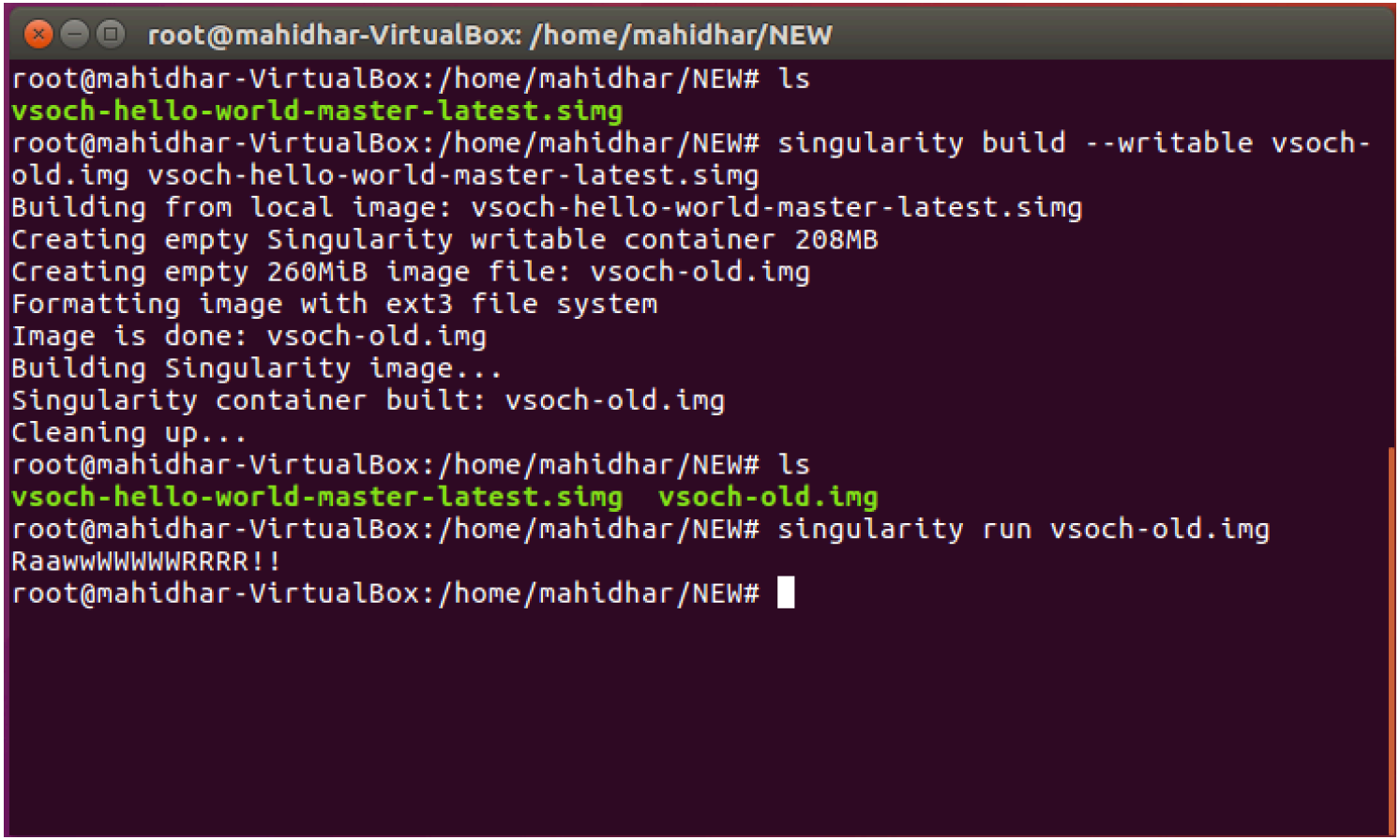
Reference: Singularity user guide: https://singularity.lbl.gov/docs-build-container

Singularity Container Image Formats

- **There are 3 different Singularity container image formats:**
  - compressed READ-ONLY **squashfs** file system suitable for production (default; *.simg file extension)
  - writable **ext3** file system suitable for interactive development (--writable option; *.img file extension)
  - writable **(ch)root directory** called a sandbox for interactive development (--sandbox option)

Build by converting existing image

- Example: On a system with root/sudo access, we can convert the image we just pulled.



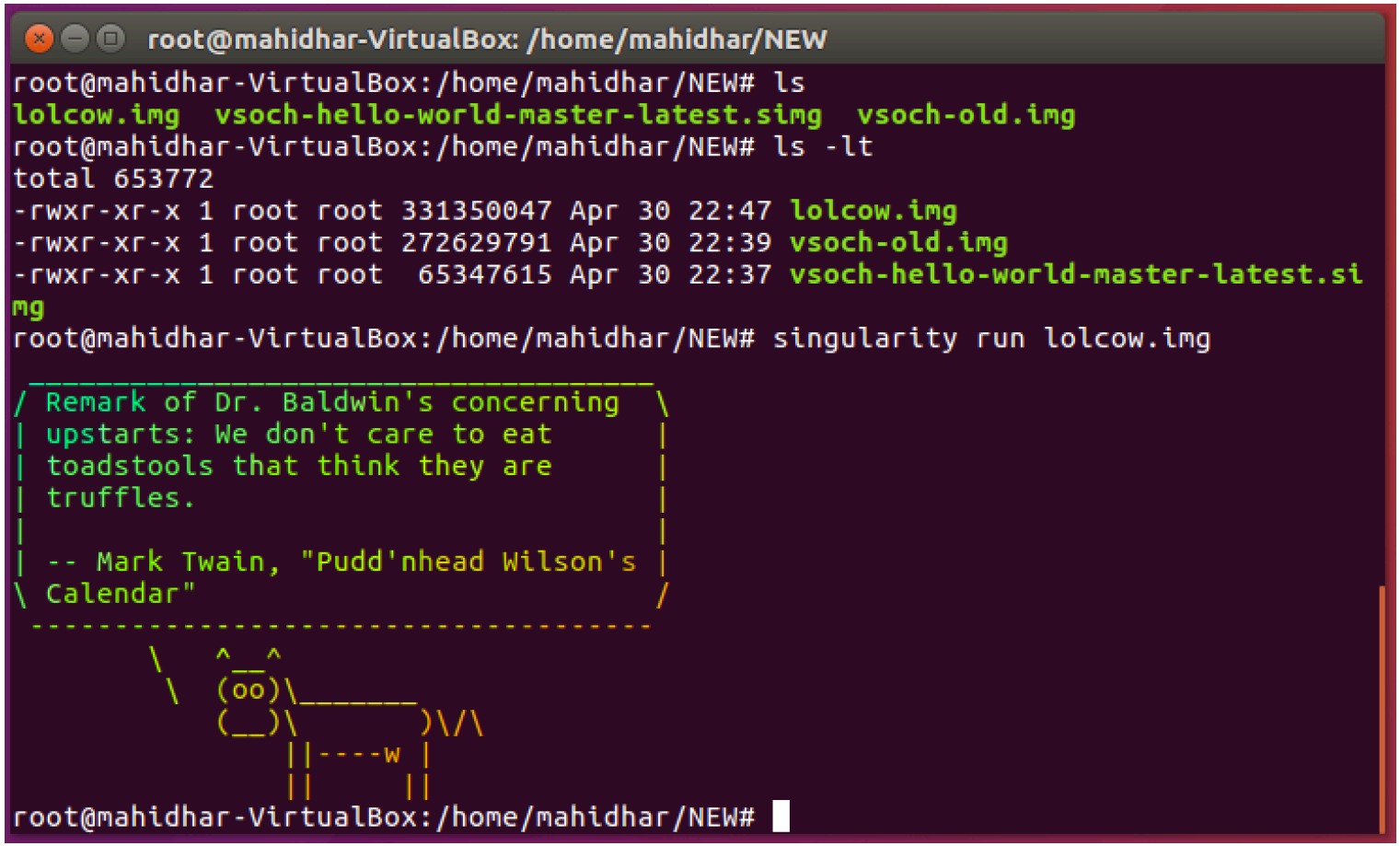
Build from docker image

singularity build --writable lolcow.img docker://godlovedc/lolcow

```
root@mahidhar-VirtualBox: /home/mahidhar/NEW

root@mahidhar-VirtualBox:/home/mahidhar/NEW# singularity build --writable lolcow
.img docker://godlovedc/lolcow
Docker image path: index.docker.io/godlovedc/lolcow:latest
Cache folder set to /root/.singularity/docker
Importing: base Singularity environment
Importing: /root/.singularity/docker/sha256:9fb6c798fa41e509b58bccc5c29654c3ff46
48b608f5daa67c1aab6a7d02c118.tar.gz
Importing: /root/.singularity/docker/sha256:3b61febd4aefe982e0cb9c696d415137384d
1a01052b50a85aae46439e15e49a.tar.gz
Importing: /root/.singularity/docker/sha256:9d99b9777eb02b8943c0e72d7a7baec5c782
f8fd976825c9d3fb48b3101aacc2.tar.gz
Importing: /root/.singularity/docker/sha256:d010c8cf75d7eb5d2504d5ffa0d19696e8d7
45a457dd8d28ec6dd41d3763617e.tar.gz
Importing: /root/.singularity/docker/sha256:7fac07fb303e0589b9c23e6f49d5dc1ff9d6
f3c8c88cabe768b430bdb47f03a9.tar.gz
Importing: /root/.singularity/docker/sha256:8e860504ff1ee5dc7953672d128ce1e4aa4d
8e3716eb39fe710b849c64b20945.tar.gz
Importing: /root/.singularity/metadata/sha256:736a219344fbca3099ce5bd1d2dbfea74b
22b830bac0e85ecca812c2983390cd.tar.gz
Creating empty Singularity writable container 253MB
Creating empty 316MiB image file: lolcow.img
Formatting image with ext3 file system
```

Running image on build host (VirtualBox in this case)



Build from definition file (examples)

- Example: ***meep.def***
- Meep is a open-source electromagnetic equation propagation software.
- Difficult to compile in default Comet environment.

meep.def

- **Some dependency installs. For example:**

  apt-get -y install libopenblas-base
  apt-get -y install libopenblas-dev
  apt-get -y install libgmp-dev
  apt-get -y install libgsl-dev
  apt-get -y install libpng16-dev
  apt-get -y install swig
  apt-get -y install guile-2.0-dev

  …

  …

meep.def

- **Some installs from source:**

```
wget https://www.open-mpi.org/software/ompi/v1.8/downloads/openmpi-1.8.4.tar.gz
 tar -xzvf openmpi-1.8.4.tar.gz
 cd openmpi-1.8.4
 ./configure --prefix=/opt/openmpi-1.8.4
  make all install
 export PATH="/opt/openmpi-1.8.4/bin:${PATH}"
 export LD_LIBRARY_PATH="/opt/openmpi-1.8.4/lib:${LD_LIBRARY_PATH}"
```

meep.def

- Setup container environment variables

```
cd /.singularity.d/env
echo 'export PATH="/opt/openmpi-1.8.4/bin:${PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/openmpi-1.8.4/lib:${LD_LIBRARY_PATH}"' >> 90-environment.sh
echo 'export PATH="/opt/harminv-1.4.1/bin:${PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/harminv-1.4.1/lib:${LD_LIBRARY_PATH}"' >> 90-environment.sh
echo 'export PATH="/opt/libctl-4.0.0/bin:${PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/libctl-4.0.0/lib:${LD_LIBRARY_PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/zlib-1.2.11/lib:${LD_LIBRARY_PATH}"' >> 90-environment.sh
echo 'export PATH="/opt/hdf5-1.10.1/hdf5/bin:${PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/hdf5-1.10.1/hdf5/lib:${LD_LIBRARY_PATH}"' >> 90-environment.sh
echo 'export PATH="/opt/h5utils-1.13/bin:${PATH}"' >> 90-environment.sh
echo 'export PATH="/opt/fftw-3.3.7/bin:${PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/fftw-3.3.7/lib:{LD_LIBRARY_PATH}"' >> 90-environment.sh
echo 'export PATH="/opt/mpb-1.6.1/bin:${PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/mpb-1.6.1/lib:${LD_LIBRARY_PATH}"' >> 90-environment.sh
echo 'export PATH="/opt/meep-1.4.3/bin:${PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/meep-1.4.3/lib:${LD_LIBRARY_PATH}"' >> 90-environment.sh
```

Build from meep.def

```
root@mahidhar-VirtualBox: /tmp/NEW
root@mahidhar-VirtualBox:/tmp/NEW# !389
singularity build --writable meep.img ./meep.def
Using container recipe deffile: ./meep.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
I: Retrieving bash 4.3-14ubuntu1
```

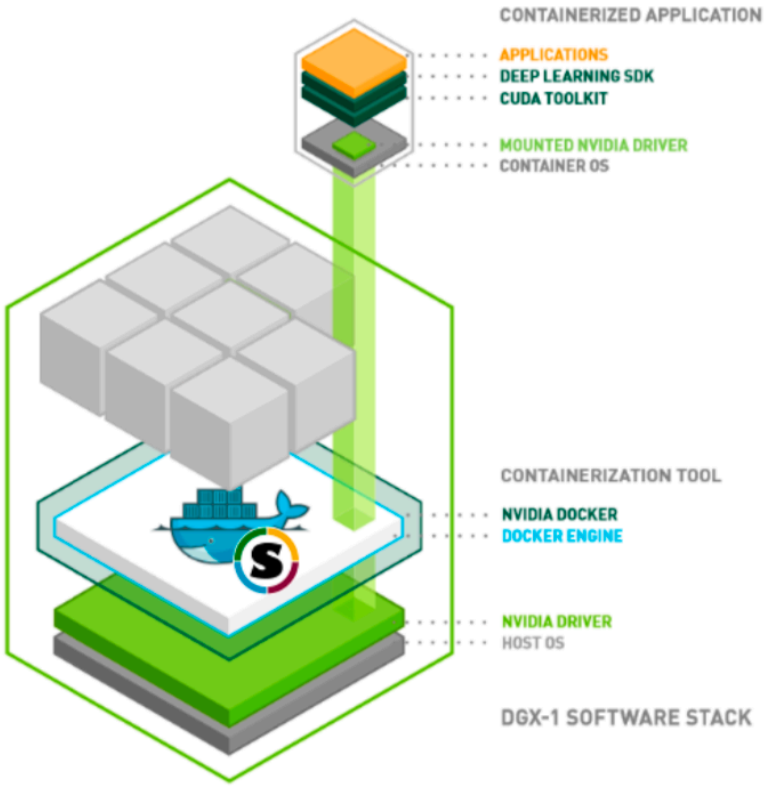
Build from meep.def

```
root@mahidhar-VirtualBox: /home/mahidhar/NEW
 /usr/bin/install -c -m 644 meep.pc '/opt/meep-1.4.3/lib/pkgconfig'
make[2]: Leaving directory '/opt/meep-1.4.3'
make[1]: Leaving directory '/opt/meep-1.4.3'
Adding deffile section labels to container
Adding runscript
Finalizing Singularity container
Calculating final size for metadata...
Skipping checks
Creating empty Singularity writable container 2251MB
Creating empty 2813MiB image file: meep.img
Formatting image with ext3 file system
Image is done: meep.img
Building Singularity image...
Singularity container built: meep.img
Cleaning up...
root@mahidhar-VirtualBox:/home/mahidhar/NEW# singularity shell meep.img
Singularity: Invoking an interactive shell within container...

Singularity meep.img:~> which meep
/opt/meep-1.4.3/bin/meep
Singularity meep.img:~> exit
exit
root@mahidhar-VirtualBox:/home/mahidhar/NEW# clear
```

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 (we do this with most of our builds).
- Today, Singularity actually allows you to bind mount the GPU driver and its supporting libraries at runtime with the --nv option.



Singularity shell

- Good for interactive tests
- Helps with compiling in the image environment.
- Very useful when the image provides the right dependencies that are required for building a particular application
- We have images with dependencies for ***Torch, Caffe, OpenCL, and Julia.***

Example of compiling using image

```
[mahidhar@comet-ln2 TUTORIAL]$ singularity shell ./ubuntu-openmpi.img
WARNING: Non existent 'bind path' source: '/scratch'
Singularity: Invoking an interactive shell within container...

Singularity ubuntu-openmpi.img:/oasis/scratch/comet/mahidhar/temp_project/Singularity/TUTORIAL>
mpif90 -o hello_mpi_ubuntu ./hello_mpi.f90

Singularity ubuntu-openmpi.img:/oasis/scratch/comet/mahidhar/temp_project/Singularity/TUTORIAL>
mpirun -np 2 ./hello_mpi_ubuntu
 node         0 : Hello world
 node         1 : Hello world

Singularity ubuntu-openmpi.img:/oasis/scratch/comet/mahidhar/temp_project/Singularity/TUTORIAL>
```

Singularity “run” command

- **Lets try on Comet:**

  module load singularity

  ls -lt /share/apps/examples/workshop/images/vsoch-old.img

  singularity run /share/apps/examples/workshop/images/vsoch-old.img

```
[mahidhar@comet-ln2 workshop]$ module li
Currently Loaded Modulefiles:
  1) intel/2013_sp1.2.144   2) mvapich2_ib/2.1        3) gnutools/2.69
[mahidhar@comet-ln2 workshop]$ module load singularity
[mahidhar@comet-ln2 workshop]$ ls -lt /share/apps/examples/workshop/images/vsoch
-old.img
-rwxr-xr-x 1 mahidhar use300 272629791 May  1 09:52 /share/apps/examples/worksho
p/images/vsoch-old.img
[mahidhar@comet-ln2 workshop]$
[mahidhar@comet-ln2 workshop]$ singularity run /share/apps/examples/workshop/ima
ges/vsoch-old.img
WARNING: Non existent bind point (directory) in container: '/oasis'
WARNING: Non existent bind point (directory) in container: '/projects'
WARNING: Non existent 'bind path' source: '/scratch'
RaawwWWWWWRRRR!!
[mahidhar@comet-ln2 workshop]$
```

Singularity exec

- Allows for command to be executed in image environment.
- This is the primary method of running in batch on Comet using the singularity images.
- Sometimes we have a simple python script and we can call a single exec command
- Can bundle workflow into a script and then execute the script via exec command.

Tensorflow via Singularity

```
#!/bin/bash
#SBATCH --job-name="TFlow_1.12"
#SBATCH --output="TFlow_1.12.%j.%N.out"
#SBATCH --partition=gpu-shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=6
#SBATCH --gres=gpu:1
#SBATCH -t 00:10:00


#Run the job
#
cp $SLURM_SUBMIT_DIR/test.py /scratch/$USER/$SLURM_JOBID
module load singularity
singularity exec /share/apps/gpu/singularity/images/tensorflow/tensorflow-gpu.si
mg python /scratch/$USER/$SLURM_JOBID/test.py
```

Tensorflow via Singularity

- **Change to the examples directory:**
  cd /home/$USER/TUTORIAL/TensorFlow

- **Submit the job:**
  sbatch TensorFlow.sb

Example with --nv option for GPUs

```
[mahidhar@comet-30-07 CUDA]$ singularity exec --nv docker://nvidia/cuda:8.0-devel-centos6 ./matrixMul
Docker image path: index.docker.io/nvidia/cuda:8.0-devel-centos6
Cache folder set to /home/mahidhar/.singularity/docker
[7/7] |===================================| 100.0%
Creating container runtime...
tar: usr/include/arpa/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/asm/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/asm-generic/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/bits/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/c++/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/drm/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/gnu/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/linux/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/mtd/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/net/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/netash/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
```

```
WARNING: Non existent bind point (directory) in container: '/oasis'
WARNING: Non existent bind point (directory) in container: '/projects'
WARNING: Non existent bind point (directory) in container: '/scratch'
WARNING: Skipping user bind, non existent bind point (file) in container: '/usr/bin/nvidia-smi'
[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "Tesla K80" with compute capability 3.7

MatrixA(320,320), MatrixB(640,320)
Computing result using CUDA Kernel...
done
Performance= 226.64 GFlop/s, Time= 0.578 msec, Size= 131072000 Ops, WorkgroupSize= 1024 threads/block
Checking computed result for correctness: Result = PASS

NOTE: The CUDA Samples are not meant for performance measurements. Results may vary when GPU Boost is enabled.
[mahidhar@comet-30-07 CUDA]$
```

Singularity and MPI

```
#!/bin/bash
#SBATCH --job-name="meep"
#SBATCH --output="meep.%j.%N.out"
#SBATCH --partition=shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=4
#SBATCH --export=ALL
#SBATCH -t 00:20:00
### Set Container to use
CONTAINER=/share/apps/examples/workshop/images/meep.img
## List the container to avoid automount issue
ls -lt /share/apps/examples/workshop/images/meep.img
## Run serial job
#time -p singularity exec ${CONTAINER} /opt/meep-1.4.3/bin/meep ./parallel-wvgs-force.ctl
## Run MPI job
module purge
module load gnutools
module load intel
module load openmpi_ib
module load singularity
time -p mpirun -np 4 singularity exec ${CONTAINER} /opt/meep-1.4.3/bin/meep ./parallel-wvgs-force.ctl
```

Multi-Node runs via Singularity

- **Easy for cases with MPI backends - we already saw this in the first talk.**

- **ML/DL frameworks can be a bit more complicated**
  - Option 1: process launches on remote nodes via ssh (need to be done via image)
  - Option 2: Use horovod (MPI based) and then MPI based launch

Commands we didn't use!

- **Image command group**
  - image.export
  - image.import
  - image.create
- **Instance command group**
  - Useful for running services like databases and web servers
  - instance.start
  - instance.list
  - instance.stop
- **Persistent overlay options**

Summary

- Singularity enables several applications on Comet.
- Examples: 1) need newer OS environment, 2) only have binaries that need specific OS, 3) import docker images with pipeline
- Can develop images on laptop and move to Comet. Persistent overlays can help do compilations on final hardware and also have data included.
- Can run multi-node jobs with MPI