

# 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

SDSC Summer Institute 2021  
Friday, August 6th, 2021  
8:30AM - 9:00AM PDT

# Today's Session

- ▶ What are containers?
- ▶ Why are containers useful?
- ▶ What is Singularity?
- ▶ How is Singularity different?
- ▶ Should you consider using Singularity?
- ▶ Q&A



# What is Containerization?



# 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

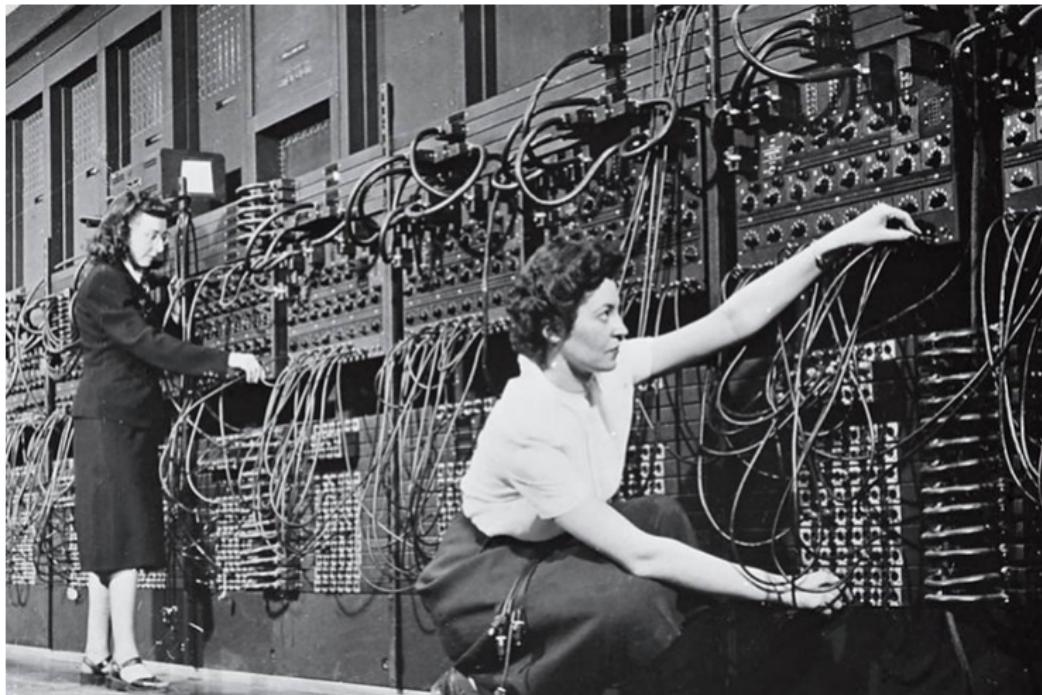


# After Containerization (Intermodal Freight Transport)

- ▶ 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



# Evolution of Software Deployment

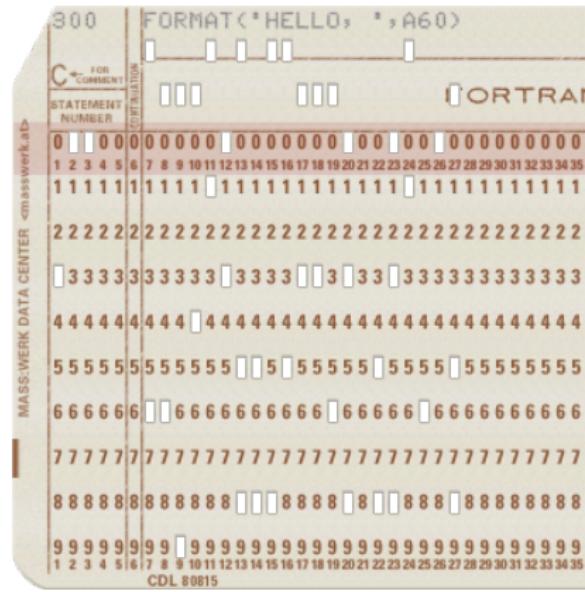


**1945:** Electronic Numerical Integrator and Computer (ENIAC)

# Evolution of Software Deployment



**1964:** IBM System/360



## Punch card

# Evolution of Software Deployment



**1998:** Desktop PC



Floppy disk



CD-ROM

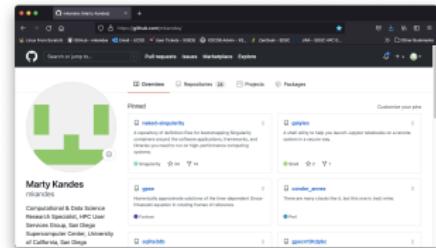
# Evolution of Software Deployment

```
SWS> aws ec2 help
~ (less)
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
```

**Today:** Amazon Web Services

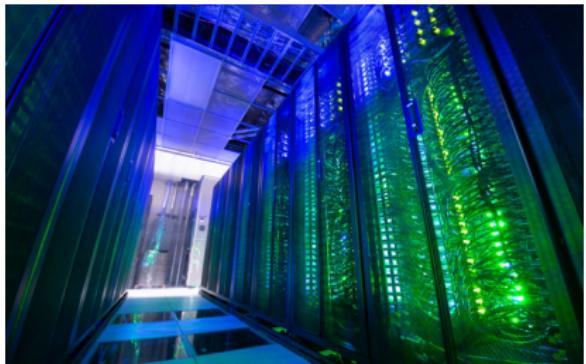


GitHub



Fiber-optics

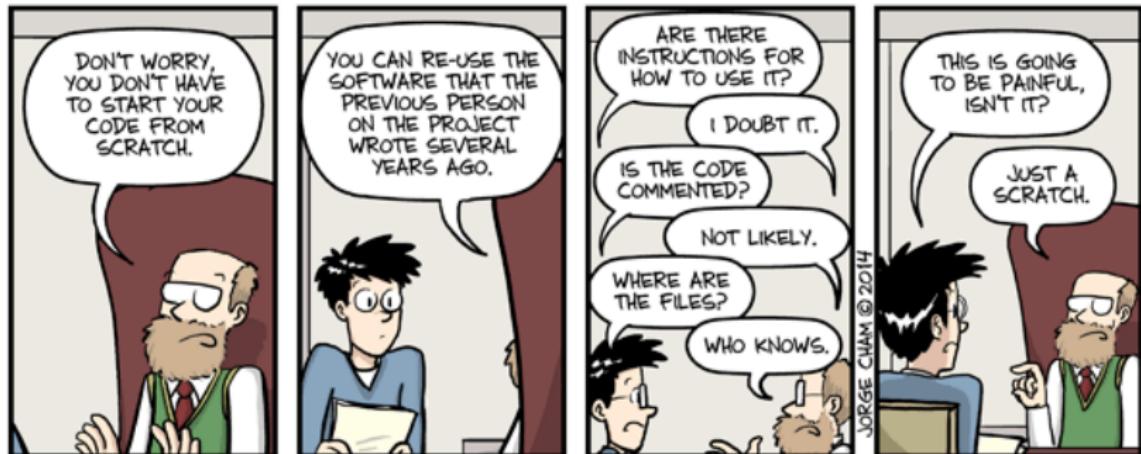
# What about deploying my software to HPC systems?



# Managing your HPC software environment

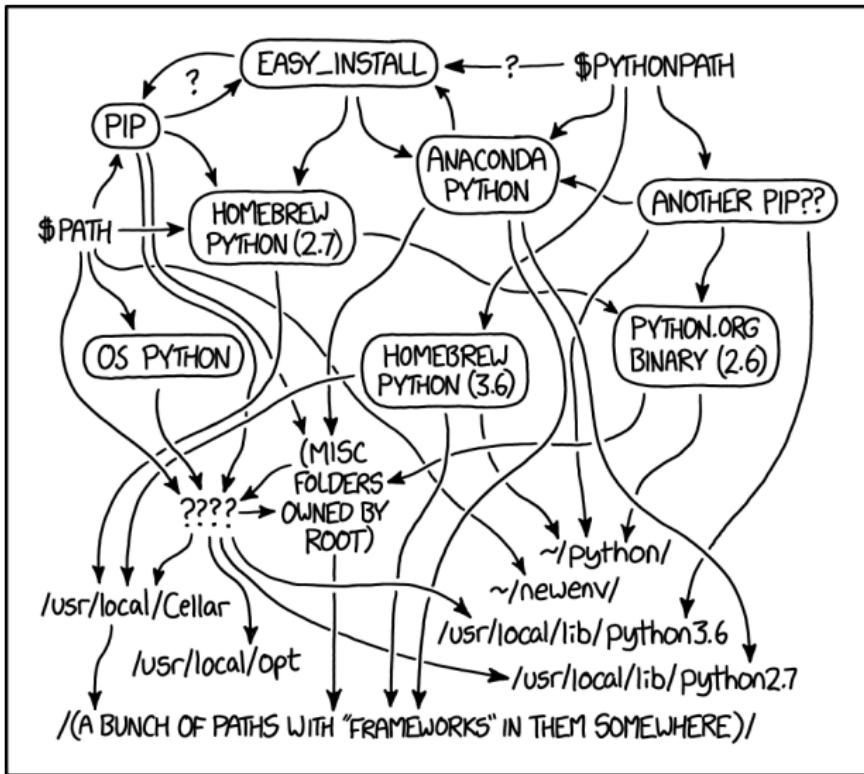


# How It Started



WWW.PHDCOMICS.COM

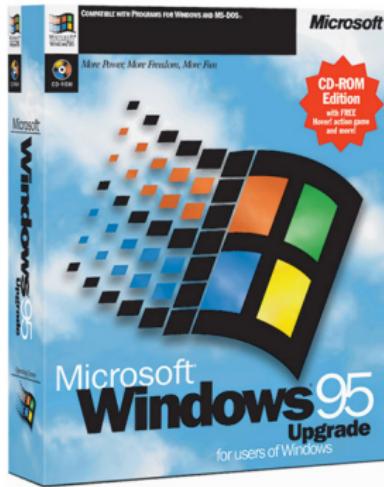
# How It's Going



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED  
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

## A Python Hodgepodge

# 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

```
mikandes@login02 ~]$ ls /cm/shared/apps/containers/singularity/
anaconda centos ciml imagemagick paraview pytorch tensorflow ubuntu visit
[mikandes@login02 ~]$ ls -lahtr /cm/shared/apps/containers/singularity/tensorflow/
total 716
-rwxr-xr-x 1 spack_gpu spack 8.8G Jan 13 2021 tensorflow-2.1.1-gpu-20200522.simg
-rwxr-xr-x 1 spack_gpu spack 9.00 Jan 13 2021 tensorflow-2.3.0-gpu-20200929.simg
-rwxr-xr-x 1 spack_gpu spack 12G Jan 13 2021 tensorflow-1.15.2-gpu-20200318.simg
-rwxr-xr-x 1 spack_gpu spack 7.6G Jan 28 2021 tensorflow-1.12-gpu-20181218.simg
-rwxr-xr-x 1 spack_gpu spack 9.7G Jun 11 18:46 tensorflow-2.3.0-20210611.sif
drwxrwxr-x 11 root spack 9 Jul 7 11:27 ..
-rwxr-xr-x 1 spack_gpu spack 17G Jul 9 13:11 tensorflow-2.5.0-ubuntu-18.04-cuda-11.2-openmpi-4.0.5-20210707.sif
lrwxrwxrwx 1 spack_gpu spack 66 Jul 9 13:18 tensorflow-latest.sif -> tensorflow-2.5.0-ubuntu-18.04-cuda-11.2-openmpi-4.0.5-20210707.sif
drwxr-xr-x 2 spack_gpu spack 8 Jul 19 07:53 .
-rwxr-xr-x 1 spack_gpu spack 7.6G Jul 19 07:53 keras-v2.2.4-tensorflow-v1.12-gpu-20190214.simg
[mikandes@login02 ~]$ ls -lahtr /cm/shared/apps/containers/pytorch/
total 276
-rwxr-xr-x 1 spack_gpu spack 8.00 Jan 27 2021 pytorch-1.5.0-gpu-20200511.simg
-rwxr-xr-x 1 spack_gpu spack 8.00 Jun 11 13:48 pytorch-1.5.0-20210611.sif
-rwxr-xr-x 1 spack_gpu spack 11G Jun 13 17:08 pytorch-1.8.1-ubuntu-18.04-cuda-11.2-openmpi-4.0.5-20210612.sif
drwxrwxr-x 11 root spack 9 Jul 7 11:27 ..
lrwxrwxrwx 1 spack_gpu spack 63 Jul 12 10:48 pytorch-latest.sif -> pytorch-1.8.1-ubuntu-18.04-cuda-11.2-openmpi-4.0.5-20210612.sif
drwxr-xr-x 2 spack_gpu spack 4 Jul 12 10:48 .
[mikandes@login02 ~]$
```

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 Definition File

A container definition file is simply a text file that contains all of the commands required to assemble a container image — i.e., 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.

```
mkandes@ubuntu:singularity-ubuntu1804-builder:~/naked-singularity/defi...$ bootstrap: debootstrap
MirrorURL: http://us.archive.ubuntu.com/ubuntu
OSVersion: focal
NLabels
APPLICATION_NAME ubuntu
APPLICATION_VERSION 20.04
APPLICATION_URL https://www.ubuntu.com

AUTHOR_NAME Marty Kandes
AUTHOR_EMAIL mkandes@sdsu.edu

LAST_UPDATED 20210223
Nsetup
Nenvironment
# Set operating system mirror URL
export MIRRORURL="http://us.archive.ubuntu.com/ubuntu"

# Set operating system version
export OSVERSION='focal'

# Set system locale
export LC_ALL='C'

# Set debian frontend interface
export DEBIAN_FRONTEND='noninteractive'

Npost -c /bin/bash
# Set operating system mirror URL
export MIRRORURL="http://us.archive.ubuntu.com/ubuntu"

# Set operating system version
export OSVERSION='Focal'

# Set system locale
export LC_ALL=C

# Set debian frontend interface
export DEBIAN_FRONTEND='noninteractive'

# 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"

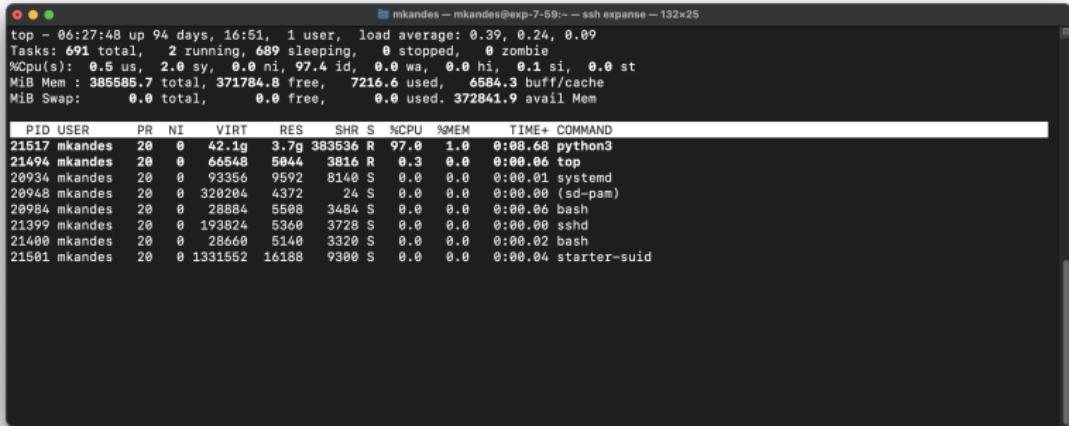
add-apt-repository -y "deb $MIRRORURL $OSVERSION-updates main"
add-apt-repository -y "deb $MIRRORURL $OSVERSION-updates universe"
add-apt-repository -y "deb $MIRRORURL $OSVERSION-updates multiverse"
add-apt-repository -y "deb $MIRRORURL $OSVERSION-updates restricted"
1,i
Top
```

# Container Process

```
mkanedes@mkanedes@login02:~/tensorflow$ ssh expance -t 132x25
[mkanedes@login02 ~]$ srun --partition=gpu-debug --account=use300 --nodes=1 --ntasks-per-node=10 --cpus-per-task=1 --mem=93G --gpus=1 --time=00:30:00 --wait=0 --pty /bin/bash
srun: job 4986810 queued and waiting for resources
srun: job 4986810 has been allocated resources
[mkanedes@exp-7-59 ~]$ cp -r /cm/shared/examples/sdsc/tensorflow/ ./
[mkanedes@exp-7-59 ~]$ cd tensorflow/
[mkanedes@exp-7-59 tensorflow]$ ls
cnn-cifar.py jupyter run-tensorflow-compute.sh run-tensorflow-gpu-shared.sh
[mkanedes@exp-7-59 tensorflow]$ module load singularitypro
[mkanedes@exp-7-59 tensorflow]$ singularity exec --nv /cm/shared/apps/containers/singularity/tensorflow/tensorflow-latest.sif python3
cnn-cifar.py
2021-08-06 06:25:33.343438: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcudart.so.11.0
2021-08-06 06:25:37.364199: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcuda.so.1
2021-08-06 06:25:37.435693: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1733] Found device 0 with properties:
pciBusID: 0000:18:00.0 name: Tesla V100-SXM2-32GB computeCapability: 7.0
coreClock: 1.53GHz coreCount: 80 deviceMemorySize: 31.75GiB deviceMemoryBandwidth: 836.37GiB/s
2021-08-06 06:25:37.435730: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcudart.so.11.0
2021-08-06 06:25:37.476310: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcUBLAS.so.11
2021-08-06 06:25:37.476381: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcUBLASLT.so.11
2021-08-06 06:25:37.490582: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libc
```

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 screenshot of a terminal window titled "mkandes - mkandes@exp-7-59:~ -- ssh expanse -- 132x25". The window displays the output of the "top" command, which shows system load statistics and a process list. The process list includes:

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
21517	mkandes	20	0	42.1g	3.7g	383536	R	97.0	1.0	0:08.68	python3
21494	mkandes	20	0	66548	5044	3816	R	0.3	0.0	0:00.06	top
20934	mkandes	20	0	93356	9592	8140	S	0.0	0.0	0:00.01	systemd
20948	mkandes	20	0	320204	4372	24	S	0.0	0.0	0:00.00	(sd-pam)
20984	mkandes	20	0	28884	5508	3484	S	0.0	0.0	0:00.06	bash
21399	mkandes	20	0	198824	5360	3728	S	0.0	0.0	0:00.00	sshd
21400	mkandes	20	0	28668	5148	3328	S	0.0	0.0	0:00.02	bash
21561	mkandes	20	0	1331552	16188	9300	S	0.0	0.0	0:00.04	starter-suid

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@exp-7-59:~ ssh expanse - 132x25
| 0 Tesla V100-SXM2... On 00000000:18:00.0 Off | 31932MiB / 32510MiB | 31% Default 0 |
| N/A 37C P0 67W / 300W | 31932MiB / 32510MiB | N/A |
+-----+
| 1 Tesla V100-SXM2... On 00000000:3B:00.0 Off | 0MiB / 32510MiB | 0% Default 0 |
| N/A 31C P0 41W / 300W | 0MiB / 32510MiB | N/A |
+-----+
| 2 Tesla V100-SXM2... On 00000000:86:00.0 Off | 0MiB / 32510MiB | 0% Default 0 |
| N/A 34C P0 40W / 300W | 0MiB / 32510MiB | N/A |
+-----+
| 3 Tesla V100-SXM2... On 00000000:AF:00.0 Off | 0MiB / 32510MiB | 0% Default 0 |
| N/A 40C P0 67W / 300W | 0MiB / 32510MiB | N/A |
+-----+
+-----+
| Processes: |
| GPU GI CI PID Type Process name GPU Memory |
| ID ID ID | Usage |
|=====|
| 0 N/A N/A 21517 C python3 31929MiB |
+-----+
```

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

```
mikandes@mkandes@login02:~/tensorflow$ ssh expanse -t 132x25
2021-08-06 06:27:53.971288: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcublasLt.so.11
1563/1563 [=====] - 11s 3ms/step - loss: 1.5196 - accuracy: 0.4463 - val_loss: 1.2614 - val_accuracy: 0.5462
Epoch 2/10
1563/1563 [=====] - 4s 3ms/step - loss: 1.1697 - accuracy: 0.5872 - val_loss: 1.1254 - val_accuracy: 0.5993
Epoch 3/10
1563/1563 [=====] - 4s 3ms/step - loss: 1.0187 - accuracy: 0.6407 - val_loss: 1.0520 - val_accuracy: 0.6389
Epoch 4/10
1563/1563 [=====] - 4s 3ms/step - loss: 0.9216 - accuracy: 0.6766 - val_loss: 0.9460 - val_accuracy: 0.6742
Epoch 5/10
1563/1563 [=====] - 4s 3ms/step - loss: 0.8554 - accuracy: 0.7007 - val_loss: 0.9213 - val_accuracy: 0.6796
Epoch 6/10
1563/1563 [=====] - 4s 3ms/step - loss: 0.7920 - accuracy: 0.7231 - val_loss: 0.9217 - val_accuracy: 0.6882
Epoch 7/10
1563/1563 [=====] - 4s 3ms/step - loss: 0.7428 - accuracy: 0.7397 - val_loss: 0.8705 - val_accuracy: 0.7017
Epoch 8/10
1563/1563 [=====] - 4s 3ms/step - loss: 0.6972 - accuracy: 0.7557 - val_loss: 0.8590 - val_accuracy: 0.7065
Epoch 9/10
1563/1563 [=====] - 4s 3ms/step - loss: 0.6528 - accuracy: 0.7704 - val_loss: 0.8864 - val_accuracy: 0.6999
Epoch 10/10
1563/1563 [=====] - 4s 3ms/step - loss: 0.6188 - accuracy: 0.7826 - val_loss: 0.8784 - val_accuracy: 0.7109
313/313 - 0s - loss: 0.8784 - accuracy: 0.7109
0.7109000086784363
[mkandes@exp-7-59 tensorflow]$
```

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.

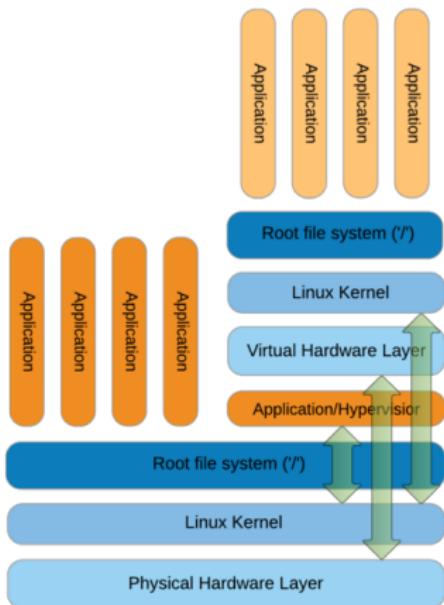
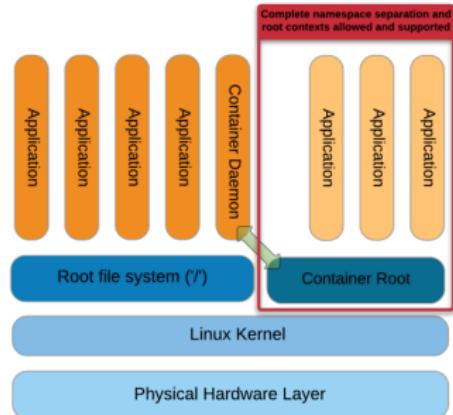
# Container : Supercomputer :: Construct : Matrix

“ ... 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

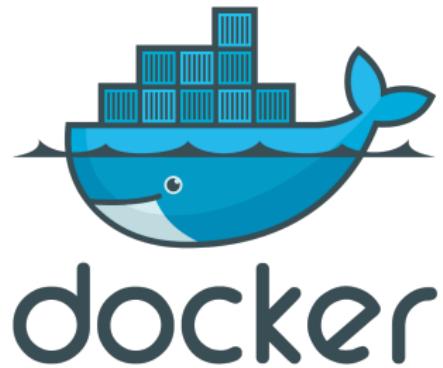
- ▶ **Performance:** Near-native application performance
- ▶ **Freedom:** Bring your own software environment
- ▶ **Reproducibility:** Package complex software applications into easy to manage, verifiable software units
- ▶ **Compatibility:** Built on open standards available in all major Linux distributions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

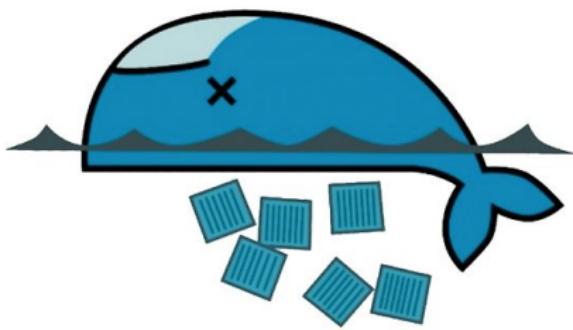
- ▶ Most common container platform in use today
- ▶ Provides tools and utilities to create, maintain, distribute, and run containers images
- ▶ Designed to accommodate network-centric services (web servers, databases, etc)
- ▶ Easy to install, well-documented, and large, well-developed user community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

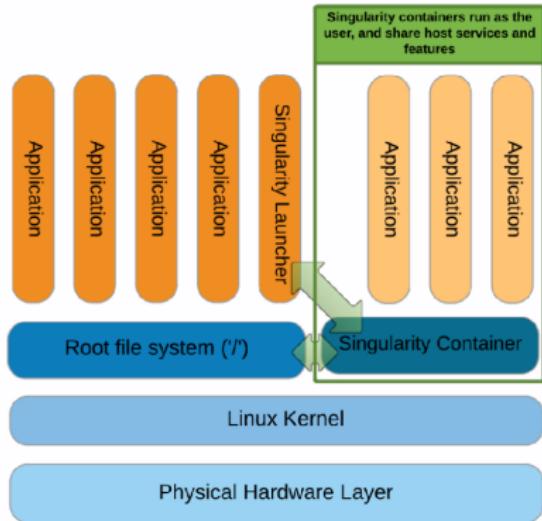


[https://github.com/hpcng/  
singularity](https://github.com/hpcng/singularity)

<https://www.sylabs.io>

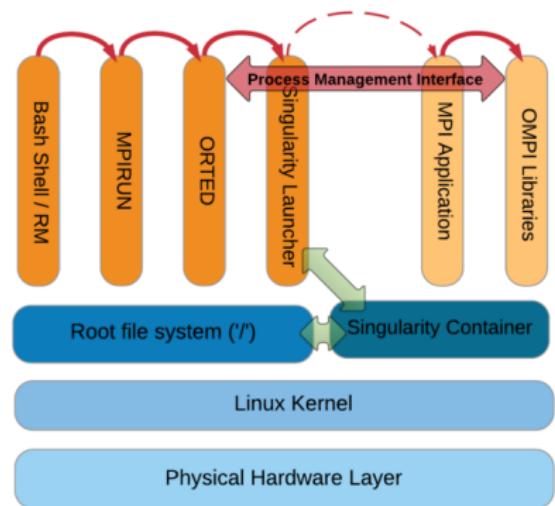
# Features of Singularity

- ▶ 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



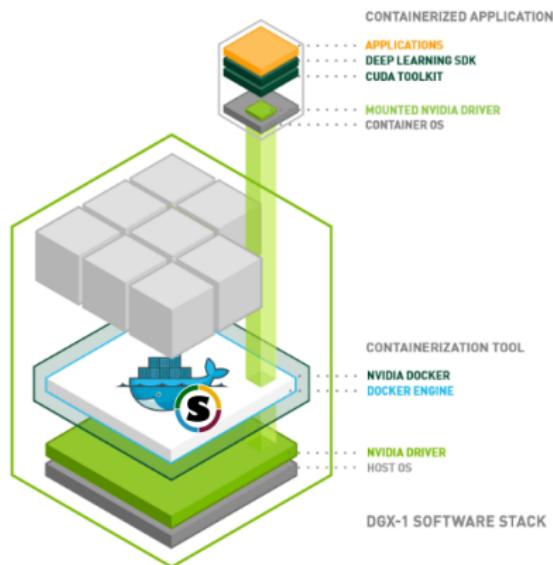
# MPI-based Singularity Containers

- ▶ 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.



# 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.



## 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



2. **Transfer** your Singularity containers to the HPC system(s) where you want to run them



3. **Run** your Singularity containers on those HPC system(s)



# 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.

## 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>

# 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>