Introduction to Containers

Gregory M. Kurtzer

@SylabsIO
@SingularityApp

Sylabs.io

Introductions...

Gregory M. Kurtzer
CEO and Founder, Sylabs Inc.

Previously spent ~20 years at LBNL/DOE as their HPC Systems Architect

Well known for founding some open source projects such as Warewulf, CentOS Linux, and most recently, Singularity.

General Overview of Linux

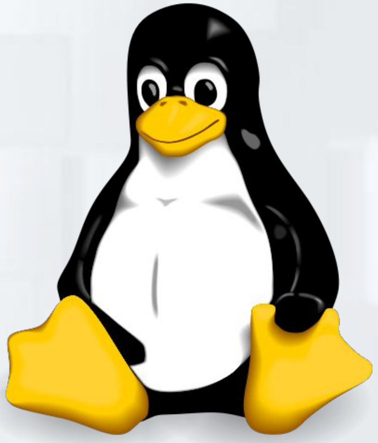
The Kernel

The Linux kernel was created by Linus Torvalds and released as an open source project in the summer of 1991.

Ker·nel - /ˈkərnl/

noun: the central or most important part of something;

“The Linux kernel is the interface between the hardware and the runtime”



Linux Distributions

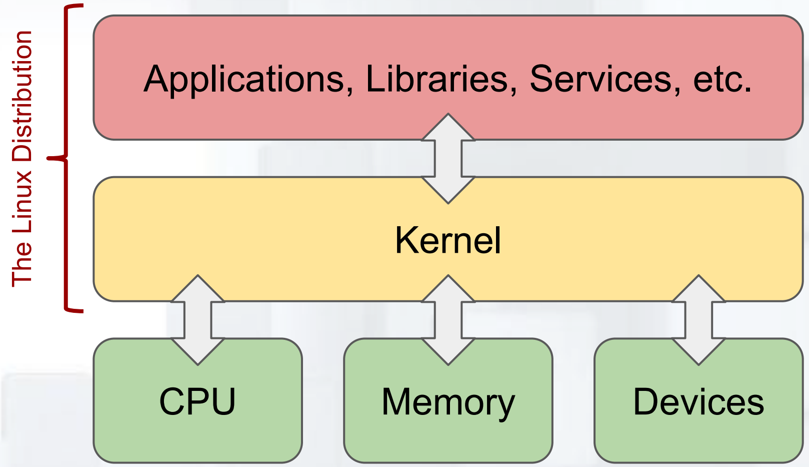
A “Linux Distribution” is the collection of applications, libraries, services, and interfaces that all run on top of the Linux kernel.

The Linux Stack

The kernel provides the interface layer between the hardware and software.

Applications, libraries, services, user interfaces, etc., all sit on top of the kernel in the "user space".

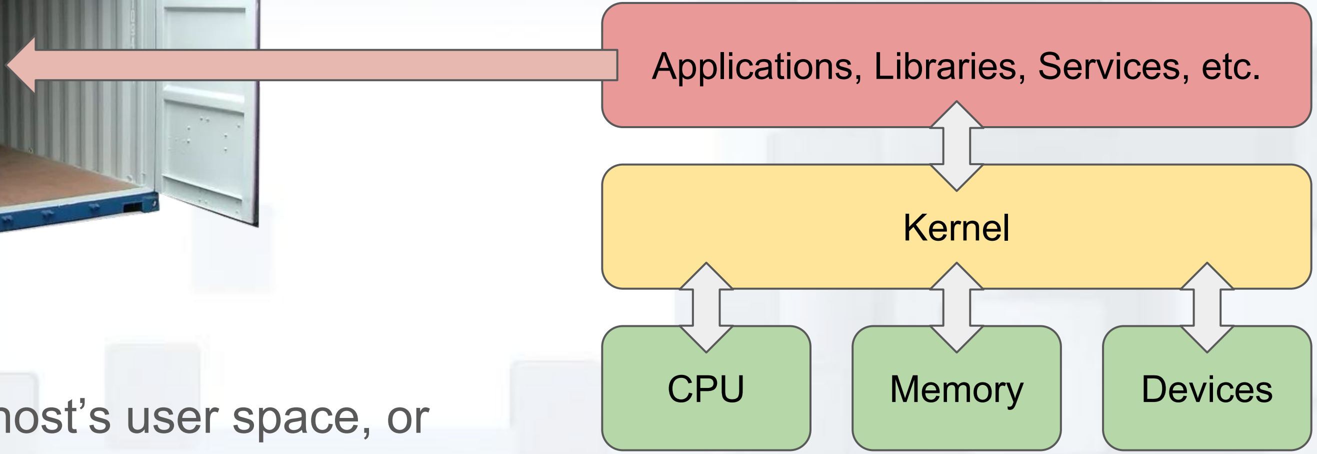
The combination of both the "kernel space" and the "user space" are provided by the Linux Distribution.



Containers in a Nutshell

What are containers?

Containers are entire encapsulations of the software stack (not including kernel).



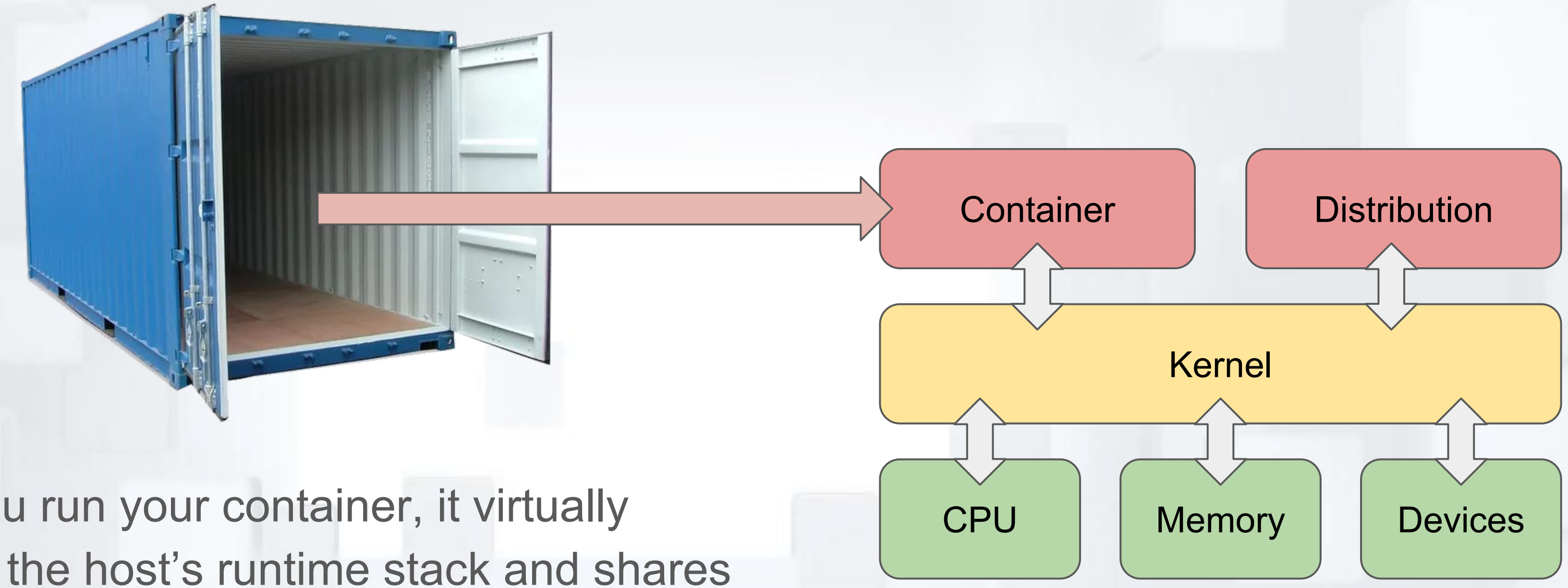
We can either use the host's user space, or we can prescriptively create one that is independent from the host.

Sylabs.io

Complete Encapsulation

Inside this container is the entire user space software environment.

Running Your Container



When you run your container, it virtually replaces the host's runtime stack and shares the host's kernel, so it is very performant and reproducible.

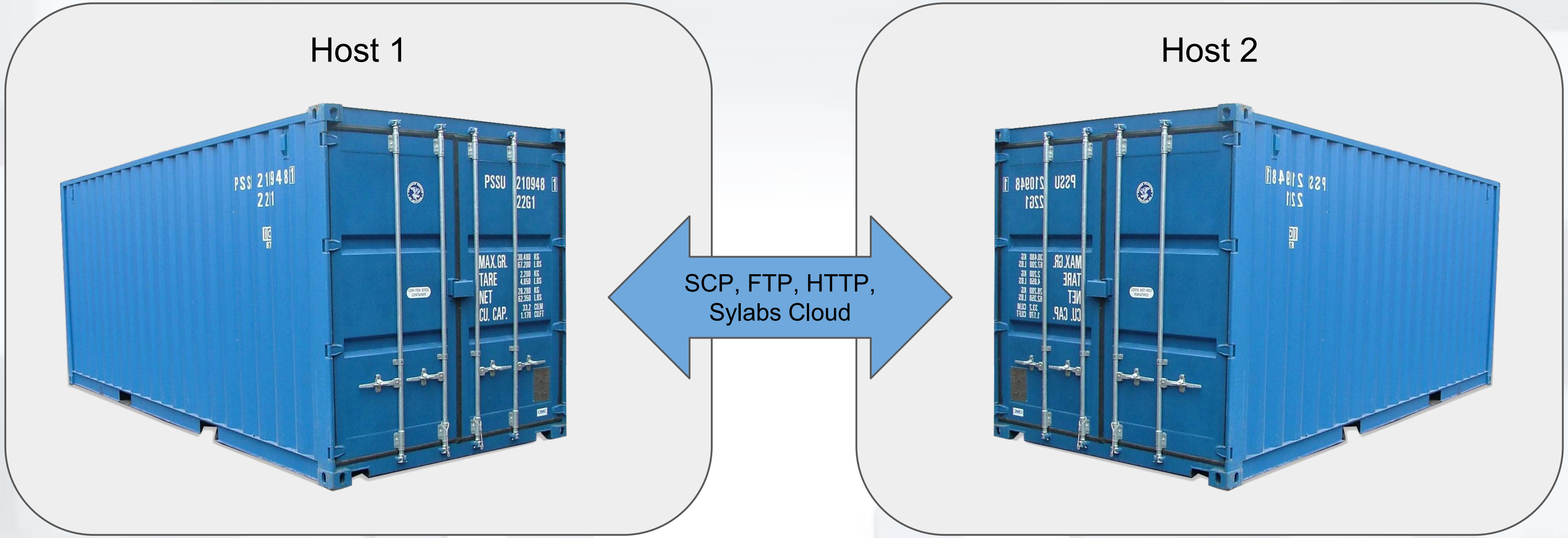
Reproducing Environments?

To duplicate or share work, one needs access to the exact same environment. If that environment is not available, it must be reproduced.

This requires all of the original components as well as a complete recipe of steps necessary to reproduce the same outcome.

Is it always possible? How do you know it is equivalent? Is it good enough?

Reproducibility and Mobility



You can run the container on different hosts, facilitating reproducibility and mobility.

Containers are all the Rage!

- Containers have changed the distribution paradigm for software
- Makes it easy to leverage other people’s work
- Creates reproducible software environments
- Empowers the end-user
- Similar like a VM without the performance impact
- Facilitates microservice based workloads by integrating into orchestration

About Singularity

Problem Statement

Scientists needed High Performance Computing (HPC) centers worldwide to support this new technology called containerization to meet their needs of **reproducibility, mobility, and freedom**

But **existing container systems fall short** as their use cases were designed for root owned microservice based workflows; security and **usage models are incompatible** with HPC

A Brief History of Time

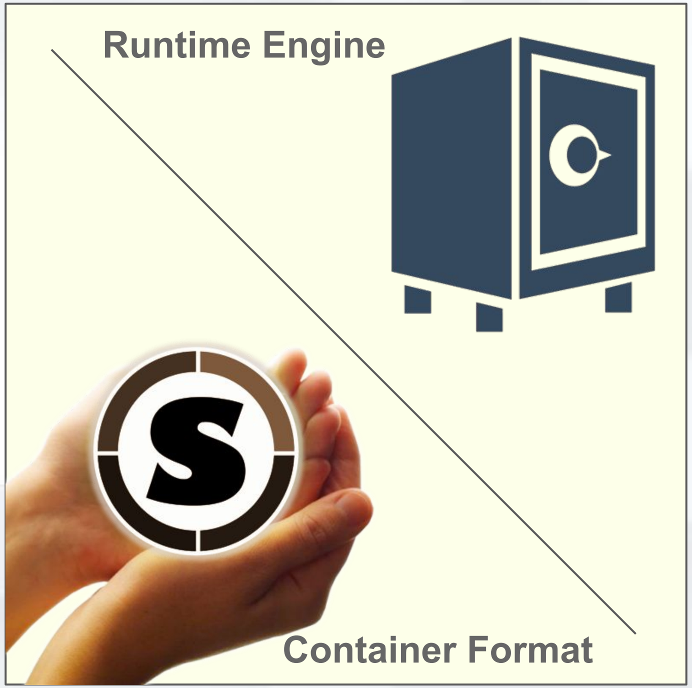
- Development of Singularity started in October, 2015
- First release (version 1.0) April 14th, 2016
- Feedback quickly demonstrated that it needed a change in design
  - Changes included a different container image format thus warranted a new major version
  - 1.x was very short lived
- With lots of help, 2.0 was developed quickly and released on June 1st, 2016
- 3.0.0 was just recently released as stable, with massive lists of new features!

Designed for Security, Mobility, and Performance

Singularity is differentiated by two primary categories:

- **Runtime Engine**: Designed to integrate when running compute based workloads, isolate when running service workloads, support security and workflow needs of HPC resources while enabling the growing enterprise cloud tools including like: Kubernetes, Mesos, Kubeflow and Docker/OCI.

- **Container Format**: The Singularity image format builds completely reproducible environments, trusted software stacks, and optimized for compute based workloads.



Adoption Skyrocketed Worldwide

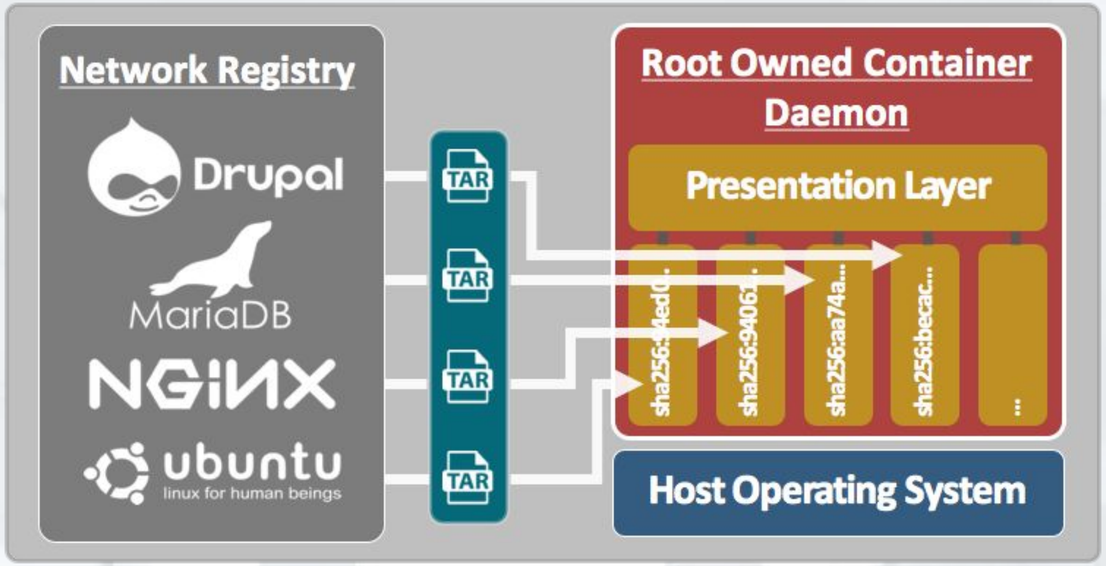
- **Summit** @ Oak Ridge National Lab
  The fastest supercomputer in the world, designed from the ground up by IBM and Nvidia specifically to run AI applications and workloads via Singularity
- **Sierra** & **Sequoia** @ Lawrence Livermore National Lab (still confirming)
- **ABCI** @ AIST
- **Titan** @ Oak Ridge National Lab
- **Stampede** & **Stampede2** @ TACC
- **Theta** @ Argonne National Lab
- **Astra** (ARM!) @ Sandia National Lab
- **MareNostrum** @ Barcelona Supercomputing Center
- **Comet** & **Gordon** @ San Diego Supercomputing Center
- **Sherlock** @ Stanford

Hewlett Packard Enterprise
Department of Defense
Kungl Tekniska Högskolan
NYU
DELL
GSI
Pfizer
USC University of Southern California Information Sciences Institute
American Museum of Natural History
NIH National Institutes of Health
NVIDIA
Eberhard Karls Universität Tübingen
Dartmouth
Fraunhofer ITWM Institut Techno- und Wirtschaftsmathematik
Stanford University
Indiana University
Fermilab 50 Years of Discovery
Memorial Sloan Kettering Cancer Center
RStor
UiO University of Oslo
Harvard University
Berkeley Lab
Washington State University
CINECA SCAI SuperComputing Applications and Innovation
PSC
Novartis
amazon web services
Università degli Studi di Milano
Florida State University
Open Science Grid
Uppsala Universitet
Ohio Supercomputer Center An OH·TECH Consortium Member
University of Michigan
intel
Yale
Melbourne Bioinformatics
Research IT
Argonne National Laboratory
The University of Iowa
University Technology Case Western Reserve University
NSC
MARCC
WestGrid
Vanderbilt University
Queen Mary University of London
Sandia National Laboratories
U.S. Department of Energy Office of Science
compute canada calcul canada
Lenovo
UT Southwestern Medical Center BioHPC
Center for High-Performance Computing
TACC Texas Advanced Computing Center

Singularity Technical Overview

1. SIF: single file container format (cryptographically verifiable and extendable runtime format)
2. No persistent global daemon processes
3. Supports non-root users running containers as themselves
4. Blocks privilege escalation within the container
5. “Bring Your Own Environment” (BYOE) usage model
6. Supports HPC workflows and architectures (MPI, resource managers, InfiniBand, FSs, etc.)
7. Supports GPUs natively (Nvidia Cuda based and AMD GFX8/9, etc.)

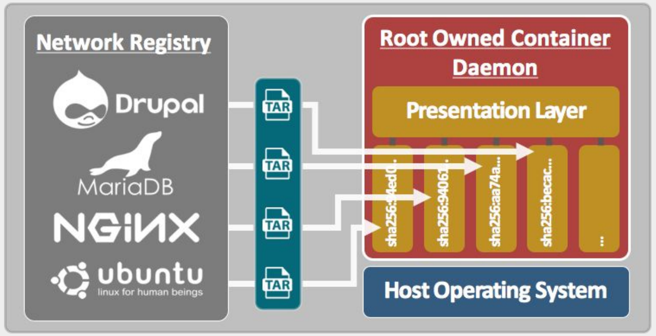
What is a container, under the hood?



Even though we metaphorically use a shipping container to describe a container, in actuality, the typical container solutions (e.g. Docker, RunC, RKT, etc.) are actually quite complicated.

Container Format Comparison

Docker, OCI, etc...



Singularity

Singularity, being a single file as well as the runtime format, simplifies the usage, administration and trust greatly!

Sylabs.io

Singularity Container Format Features

Reproducible

X = X

Archival

Mobile

Controls Compliant

Singularity Runtime Features

Accelerated GPUs

Designed For Performance

Supports GUIs

Orchestration Agnostic

MESOS

kubernetes

slurm workload manager

UNIVA

OPEN CONTAINER INITIATIVE

docker

Compatible with HPC

Jobs and Services

Standards Compliant

Secure

Sylabs.io

How You Can Make Use Of Containers With Singularity Today

Examples and use-cases

Sylabs.io

Building a Container with Singularity

```
$ cat recipe.def
Bootstrap: yum
OSVersion: 7
MirrorURL: http://mirror.centos.org/centos-%{OSVERSION}/%{OSVERSION}/os/x86_64/
Include: yum

%post
yum -y install vim-minimal

%runscript
exec /usr/bin/vi “$@”
$ singularity build centos-vim.sif recipe.def
…
```

Example Usage of Singularity

```
# You can use the container as follows...
$ singularity shell centos-vim.sif
$ singularity exec centos-vim.sif sleep 1
$ singularity run centos-vim.sif testfile.txt

# Singularity containers are also executable, so you can ‘run’ them directly
$ ./centos-vim.sif testfile.txt

# You can move the container onto any other Linux system with Singularity
installed, and use the container directly
$ scp centos-vim user@examplehost.com:
$ ssh user@examplehost.com
$ ./centos-vim newtestfile.txt
```

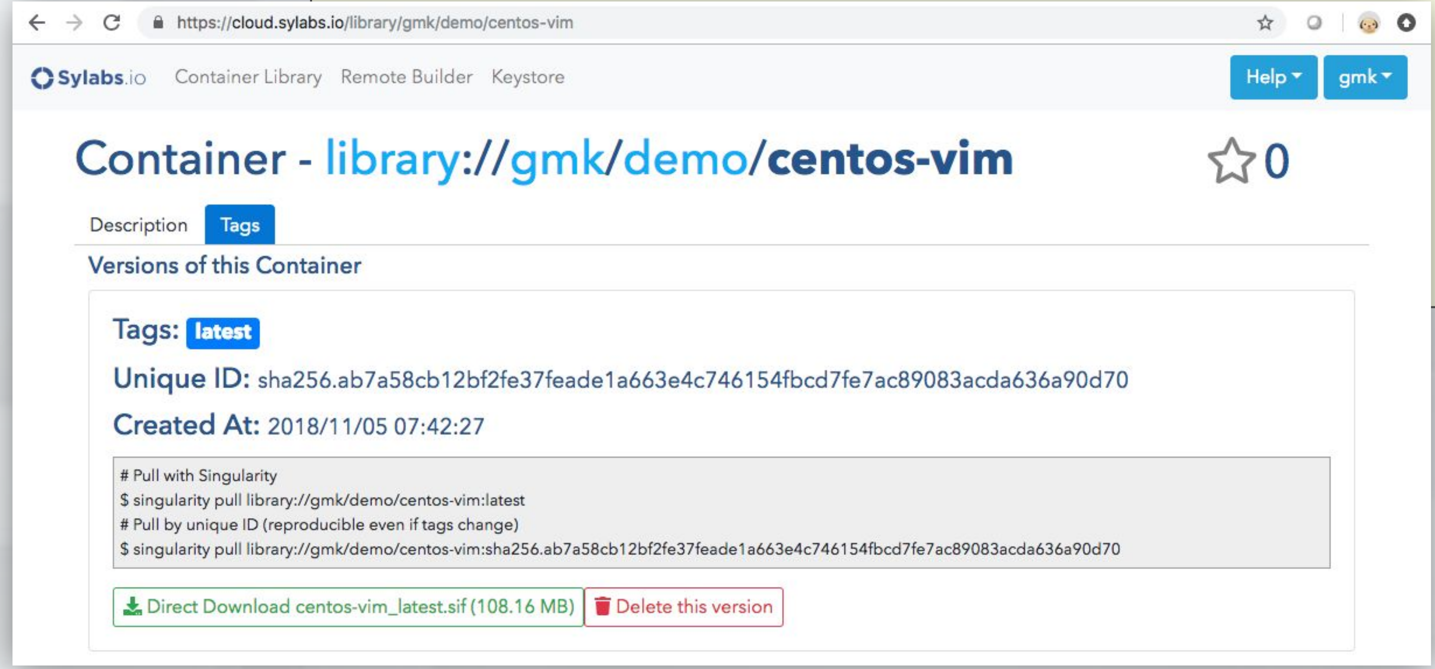
Cryptographically signed containers

```
$ singularity sign centos-vim.sif
Signing image: centos-vim.sif
No OpenPGP signing keys found, autogenerate? [Y/n] y
Enter your name (e.g., John Doe) : Greg
Enter your email address (e.g., john.doe@example.com) : g@sylabs.io
Enter optional comment (e.g., development keys) : demokeys
Generating Entity and OpenPGP Key Pair... Done
Enter encryption passphrase :
…
Uploaded key successfully!
Enter key passphrase:
Signature created and applied to centos-vim.sif
$
```

Sylabs.io

Using the Sylabs Container Library

```
$ singularity push centos-vim.sif library://gmk/demo/centos-vim:latest
INFO:    Now uploading centos-vim.sif to the library
 108.16 MiB / 108.16 MiB [==============================] 100.00% 13.75 MiB/s 7s
INFO:    Setting tag latest
$
```



Container - library://gmk/demo/centos-vim

Versions of this Container

Tags: latest

Unique ID: sha256.ab7a58cb12bf2fe37feade1a663e4c746154fbcd7fe7ac89083acda636a90d70

Created At: 2018/11/05 07:42:27

```
# Pull with Singularity
$ singularity pull library://gmk/demo/centos-vim:latest
# Pull by unique ID (reproducible even if tags change)
$ singularity pull library://gmk/demo/centos-vim:sha256.ab7a58cb12bf2fe37feade1a663e4c746154fbcd7fe7ac89083acda636a90d70
```

Direct Download centos-vim_latest.sif (108.16 MB)

Delete this version

Pulling and Validating a Container

```
$ singularity pull library://gmk/demo/centos-vim:latest
 108.16 MiB / 108.16 MiB [==============================] 100.00% 35.00 MiB/s 3s
$ singularity verify centos-vim_latest.sif
Verifying image: centos-vim_latest.sif
INFO:    key missing, searching key server for KeyID: 58D8405A30E12DE6...
INFO:    key retreived successfully!
Store new public key F56D95BD3AFAC6FA3423911A58D8405A30E12DE6? [Y/n] y
Data integrity checked, authentic and signed by:
     Greg (demokeys) <g@sylabs.io>, KeyID 58D8405A30E12DE6
$
```

Working with Singularity Hub

```
$ singularity pull shub://GodloveD/lolcow:latest
 87.57 MiB / 87.57 MiB [================================] 100.00% 80.54 MiB/s 1s
$ ./lolcow_latest.sif
 _______________________________________
/ Q: What's tiny and yellow and very,   \
| very, dangerous? A: A canary with the |
\ super-user password.                  /
 ---------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Working with Docker

```
# Building a Singularity container (SIF) from DockerHub
$ singularity build tensorflow.sif docker://tensorflow/tensorflow:latest
…
# Running a shell directly from DockerHub
$ singularity shell docker://ubuntu:latest
Singularity ubuntu_latest.sif:~/demo> cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.1 LTS"
Singularity ubuntu_latest.sif:~/demo> exit
$ singularity exec docker://centos:latest cat /etc/redhat-release
CentOS Linux release 7.5.1804 (Core)
```

Blurring the Line Between Container and Host

```
# Build a container with the latest version of Python
$ singularity build python.sif docker://python:latest
…
$ singularity exec python.sif python3 --version
Python 3.7.1
$ singularity exec python.sif python3 hello.py
Hello, World!
$ cat hello.py | singularity exec python.sif python3
Hello, World!
$ singularity shell python.sif
Singularity python.sif:~> python3 hello.py
Hello, World!
Singularity python.sif:~> exit
```

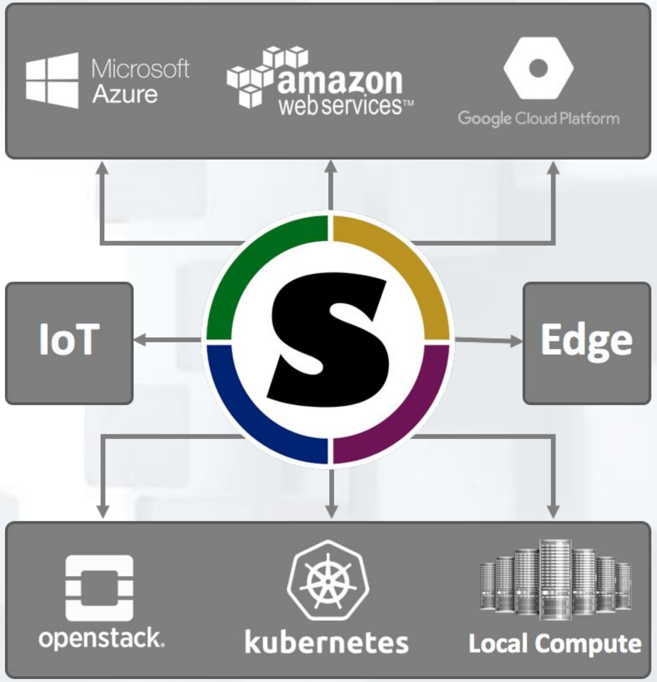
Sylabs.io

Accessing the Host's GPU With Tensorflow

```
$ singularity exec --nv docker://tensorflow/tensorflow:latest-gpu python
Python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> x1 = tf.constant([1,2,3,4])
>>> x2 = tf.constant([5,6,7,8])
>>> result = tf.multiply(x1, x2)
>>> print(result)
Tensor("Mul:0", shape=(4,), dtype=int32)
>>> exit()
$
```

In Summary...

With Singularity, you can encapsulate the entire user space runtime environment including workflows, applications, and data, all into a single file which is guaranteed to be reproducible.



Introduction to Containers

Gregory M. Kurtzer

@SylabsIO
@SingularityApp

Sylabs.io