Essential Singularity: Containers for Scientific and High-Performance Computing

Marty Kandes, Ph.D.

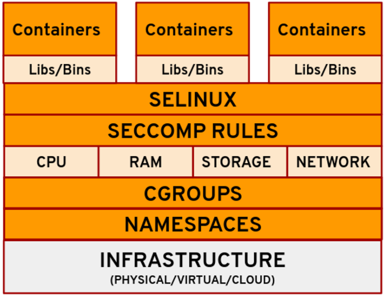
Computational & Data Science Research Specialist
High-Performance Computing User Services Group
San Diego Supercomputer Center
University of California, San Diego

CIML Summer Institute
Tuesday, June 28th, 2022
8:05 AM - 9:05 AM PT

Essential Singularity: Containers for Scientific and High-Performance Computing

- What is a (software) container?
- What is Singularity?
- What are the (three) essential `singularity` commands?

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 vs. Container Process

- 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.

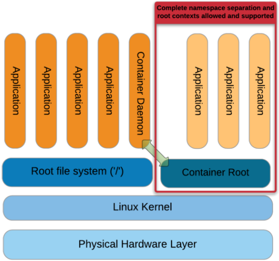
- A **container process** is simply a standard (Linux) process running on top of the underlying host's operating system and kernel, but whose software environment is defined by the contents of the *container image*.

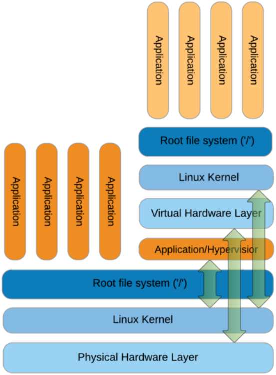
Container : Supercomputer :: Construct : Matrix

" ... it's our loading program."

" We can load anything ... anything we need."

Containers vs. Virtual Machines





Containerized applications have **direct access** to the host kernel and hardware and, thus, are able to achieve similar performance to host-native compiled and run applications.

Virtualized applications only have **indirect access** to the host kernel and hardware via the guest OS and hypervisor, which (generally) 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

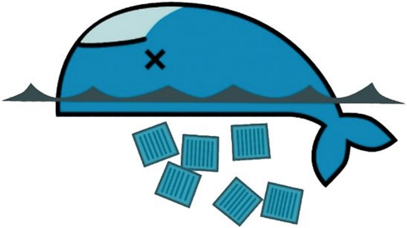
Docker

- 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

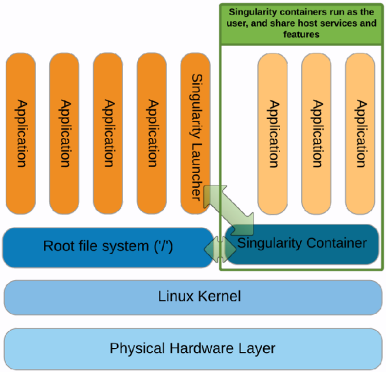
https://www.sylabs.io

Apptainer

https://apptainer.org

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

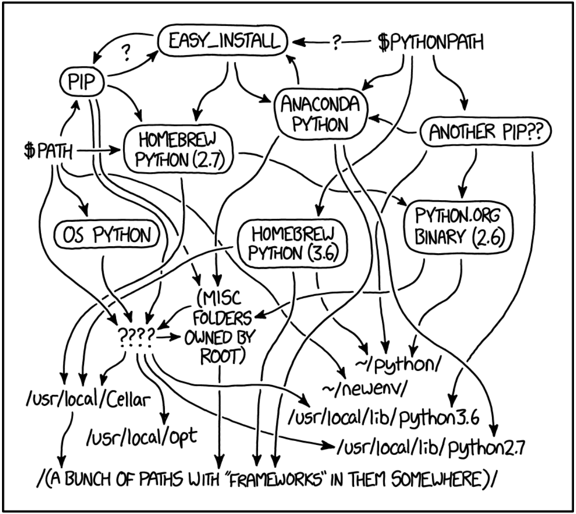
Essential Singularity

The (three) essential `singularity` commands are:

```
singularity [global options] <command> [command options] ...
```

- `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)
- `exec`: Execute an arbitrary command within your container.
- `shell`: Spawn an interactive shell session inside your container.

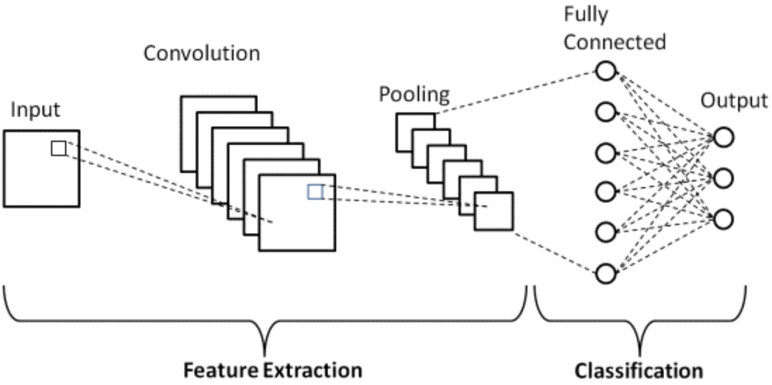
Exercise 1: python shell game



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

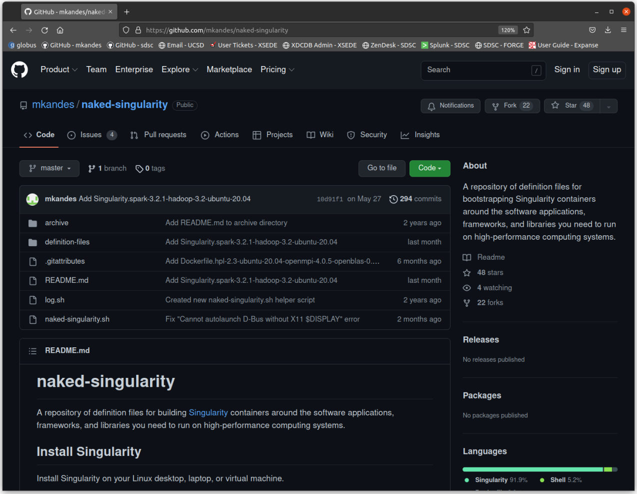
Exercise 2: bind on through (to the other side)

Exercise 3: dash dash nv to CIFAR



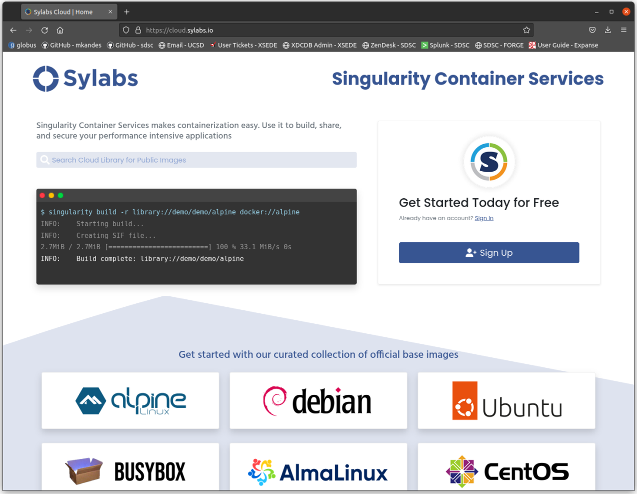
Exercise 4: Distribute this Horovod

naked-singularity



https://github.com/mkandes/naked-singularity

Extra Credit: Remote Build Service



https://cloud.sylabs.io

Singularity: A Summary

- 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.
- In many cases, your software is now completely portable between the different HPC systems you want to run on.
- 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
- **Sylabs YouTube Channel**: https://www.youtube.com/c/SylabsInc
- **Apptainer Project**: https://apptainer.org

Questions?

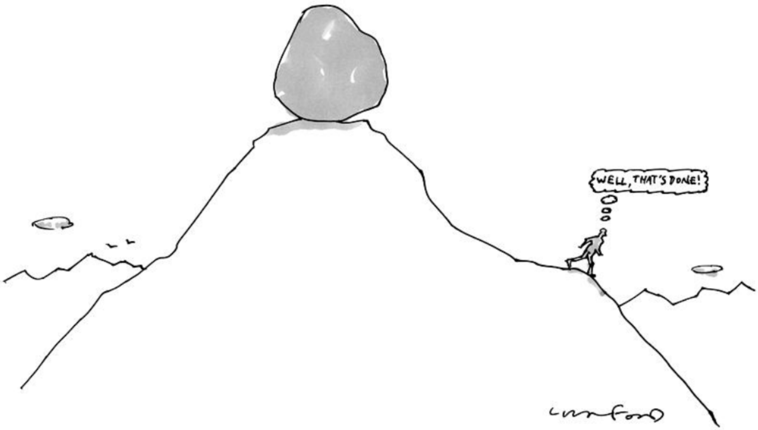


Image Credit: New Yorker - M. Crawford