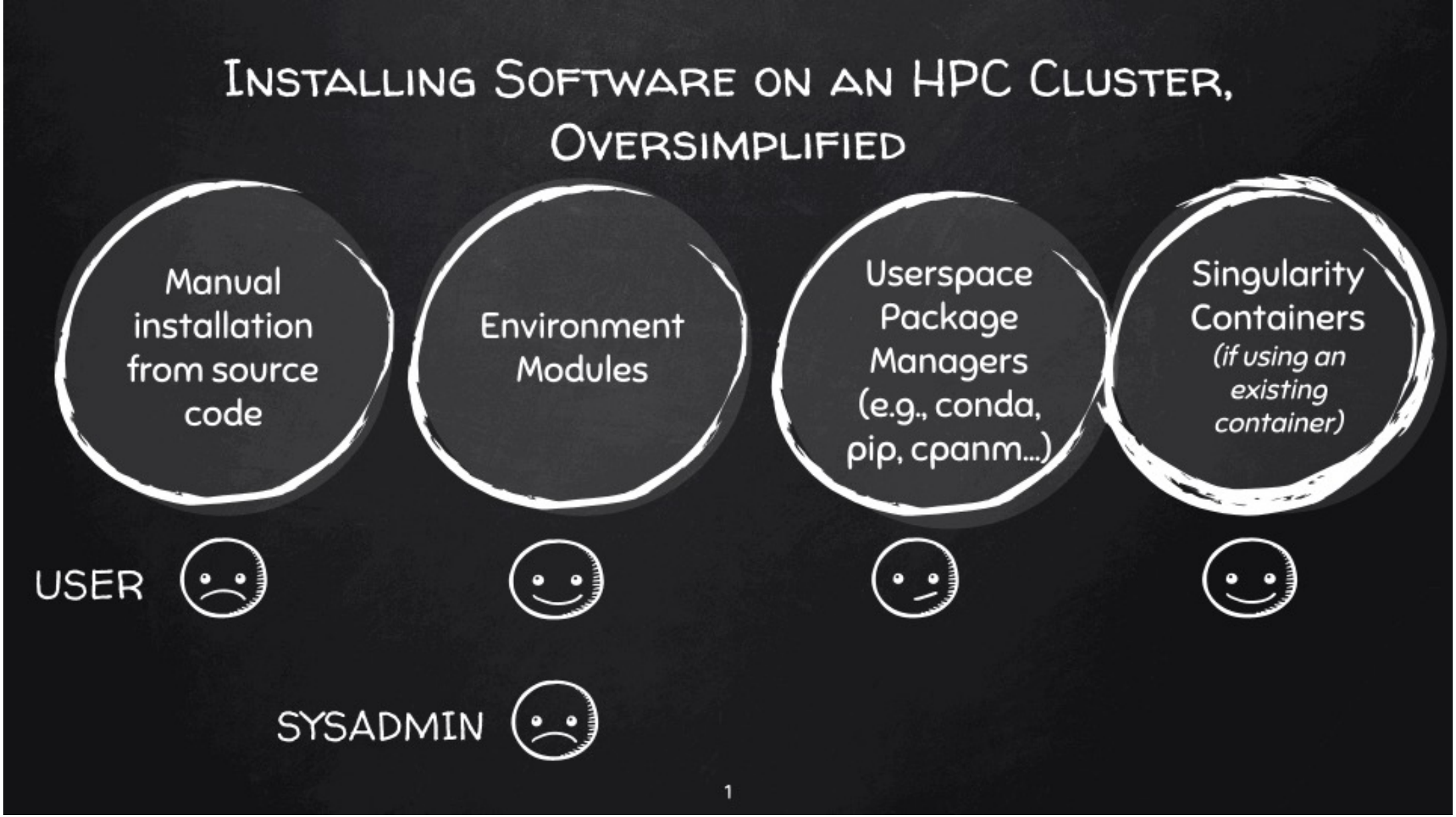
Tufts UNIVERSITY

Mastering Reproducibility: pulling, running and building containerized HPC applications

Yucheng Zhang

Bioinformatics Engineer

TTS Research Technology



Source: https://github.com/harvardinformatics/bioinformatics-coffee-hour/blob/master/singularity/images/installing_software.gif

Overview

- What are containers and why should we use them?
- How to use Singularity/Apptainer to pull and run containers?
- How to build your own containers?
- Environment modules based on containers
- How to build and run MPI and GPU containers?
- Hands-on Demo

Pulling, Running and Building Containerized HPC Applications

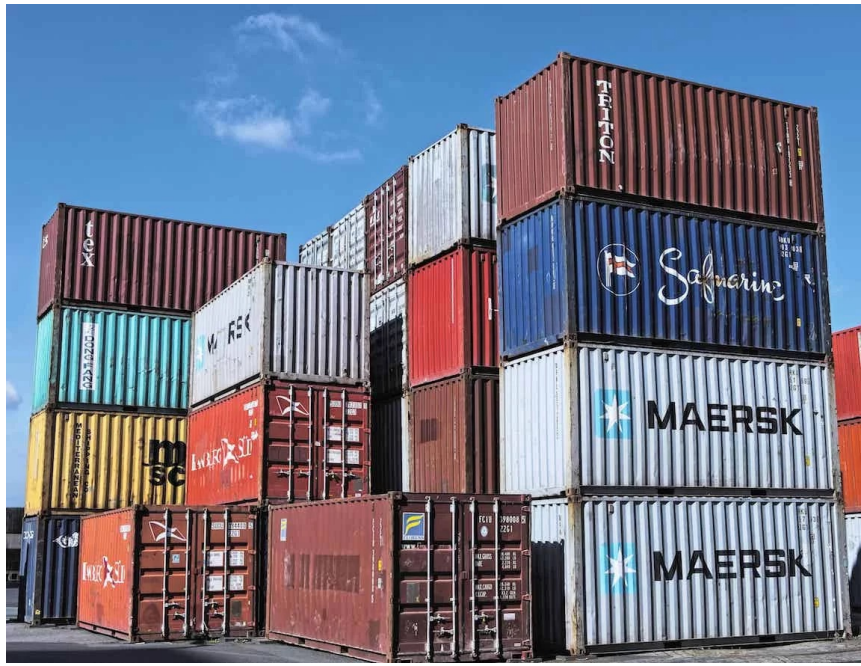
Background

Shipping containers

- **The arrival of modern shipping containers changed our transportation industry.**

- **Container is a standardized way to package items together into one shipment.**

  1. Standard packaging
  2. Isolation and efficiency
  3. Separation of concerns
  4. Portable



Containers

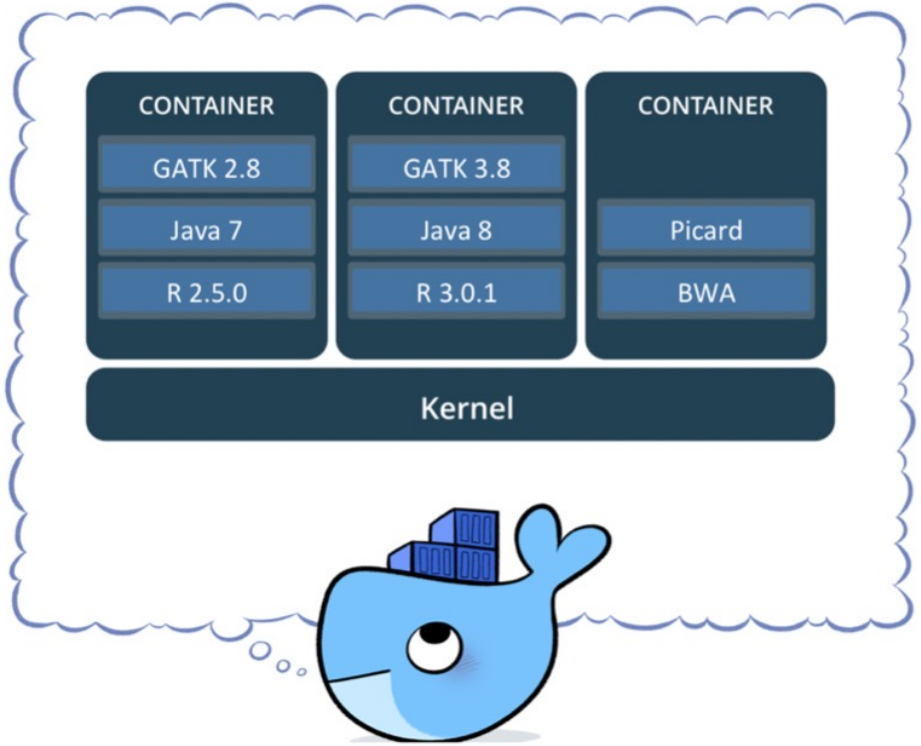
A **container** is an abstraction for a set of technologies that aim to solve the problem of how to get software to run reliably when moved from one computing environment to another.

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.

Registry: a place to store (and share) container images.

Why use containers?

- Getting organized: containers keep things organized by isolating programs and their dependencies inside containers.
- Build once, run **almost** anywhere: containers allow us to package up our complete software environment and ship it to other computing environments.
- Reproducibility: containers can ensure identical versions of apps, libraries, compliers, etc.



NATURE | NEWS

Over half of psychology studies fail reproducibility test

Largest replication study to date casts doubt on many published positive results.

SEPTEMBER 20, 2013

Science is in a reproducibility crisis: How do we resolve it?

by Fiona Fidler & Ascelin Gordon, The Conversation

News Feature | Published: 25 May 2016

1,500 scientists lift the lid on reproducibility

Monya Baker

Nature **533**, 452–454 (2016) | Cite this article

Gregory Kurtzer • Following
Founder of Rocky Linux, the RESF, and the CEO of CIQ
8h •

This is why I love open source "freeloaders" and why I do what I do.

Open source software (which I created -- Apptainer/Singularity) is enabling research like this. The same goes for Warewulf, Rocky Linux (and CentOS previously), and everything we are doing at CIQ.

https://lnkd.in/gw-3NVaw

ALZHEIMER'S ASSOCIATION | Alzheimer's & Dementia®
THE JOURNAL OF THE ALZHEIMER'S ASSOCIATION

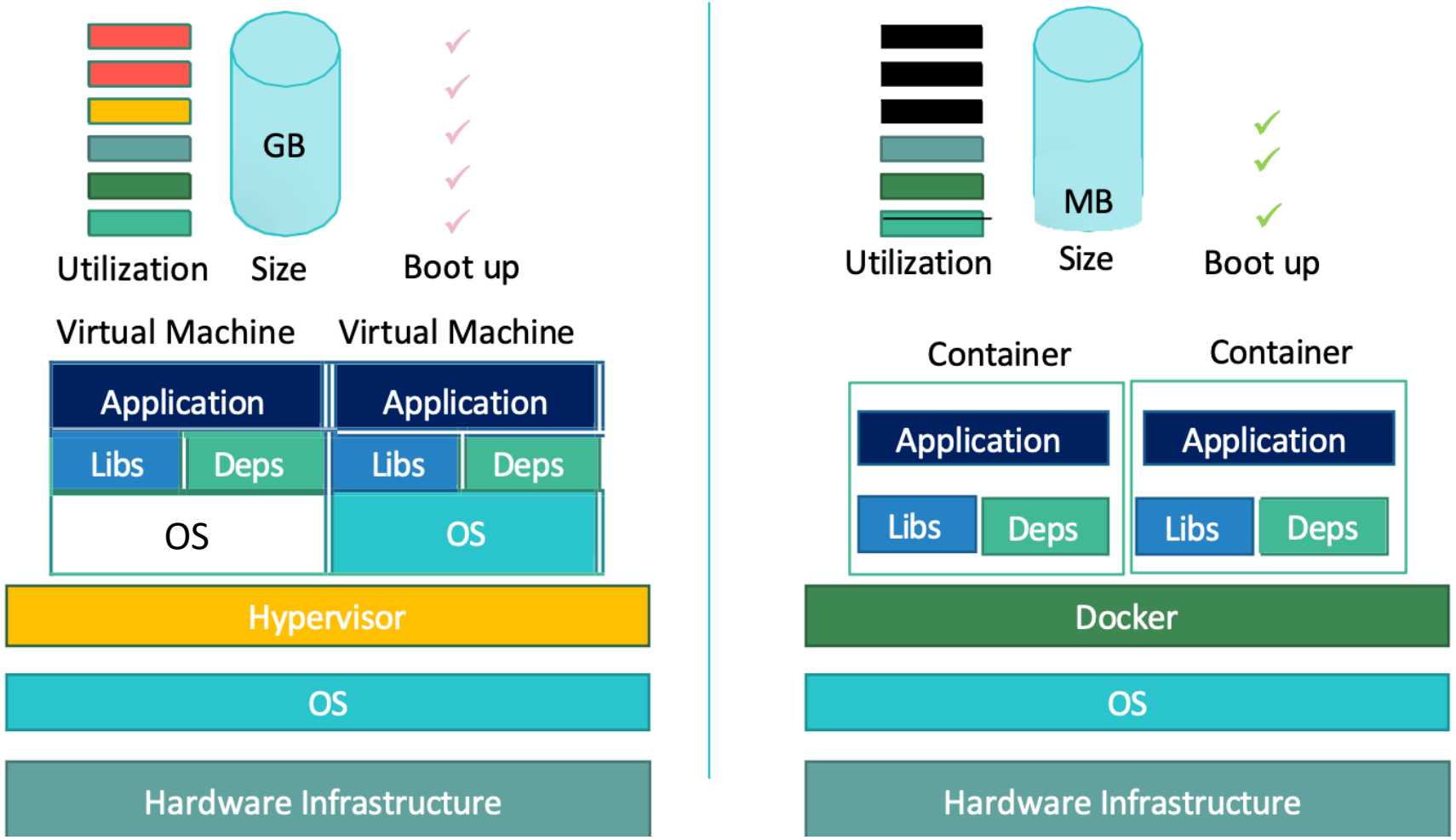
ALZHEIMER'S IMAGING CONSORTIUM | Free Access

Containerizing neuroimaging workflows for scalable and reproducible analyses

Sarah J. Keefe, Nicole S. McKay, Pamela J. LaMontagne, Shaney Flores, Brian A. Gordon, Tammie L.S. Benzinger

First published: 24 December 2023 | https://doi.org/10.1002/alz.081785

Containers vs Virtual Machines

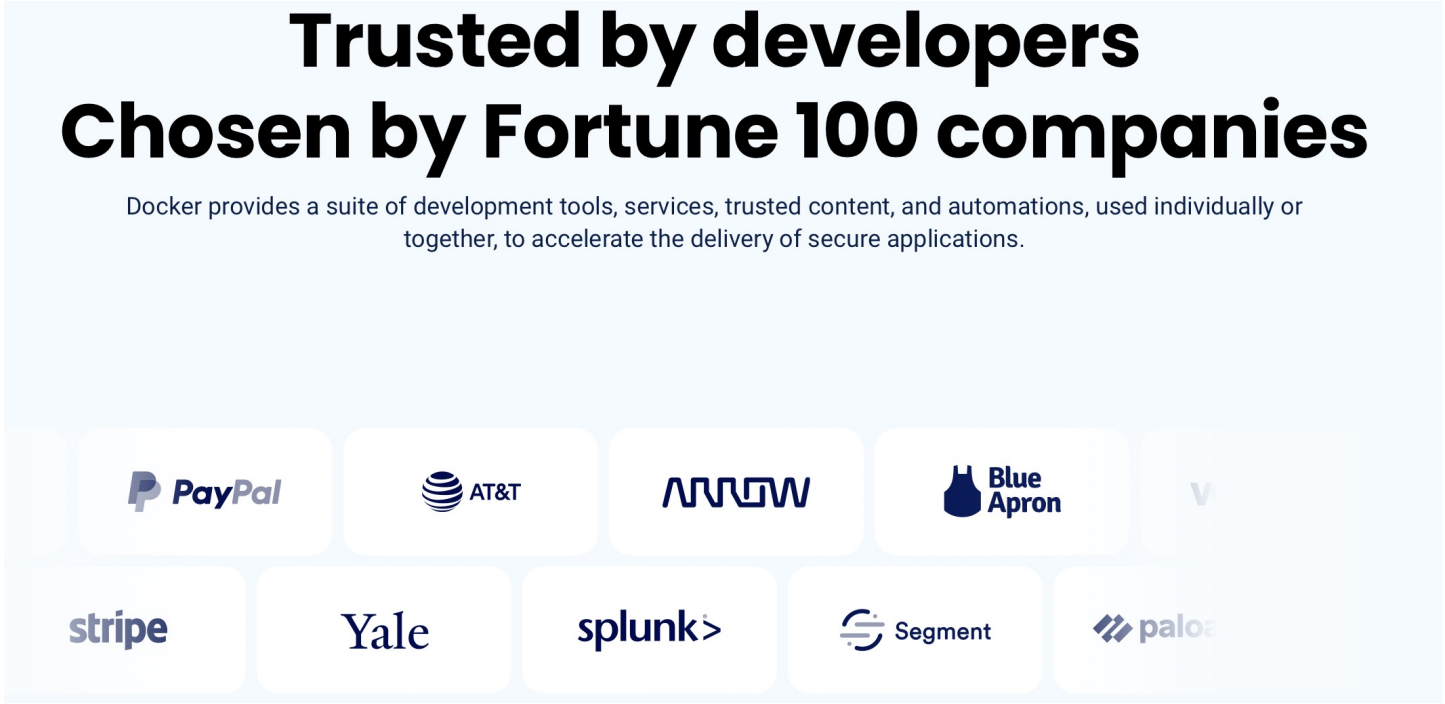


https://kodekloud.com/courses/kubernetes-for-the-absolute-beginners-hands-on/

Docker

The concept of containers emerged in 1970s, but they were not well known until the emergence of Docker containers in 2013.

Docker is an open source platform for building, deploying, and managing containerized applications.



Why not Docker on HPC?

Summarized by Stanford HPC center

- Docker requires a **daemon** running as **root** on all compute nodes, which poses security risks.
- All authenticated actions, such as login and push, execute as root. Therefore, those functions cannot be used simultaneously by multiple users on the same node.
- Docker uses **cgroups** to isolate containers, as does the Slurm scheduler, which uses **cgroups** to allocate resources to jobs and enforce limits. Those uses are unfortunately conflicting.
- Most importantly, ***allowing users to run Docker containers will give them* root *privileges* inside that container, thereby enabling them to access any of the filesystems on the cluster as root**. This opens the door to user impersonation, inappropriate file tampering or stealing.

https://www.sherlock.stanford.edu/docs/software/using/singularity/#why-not-docker

Singularity

- Singularity was developed in 2015 as an open-source project by researchers at Lawrence Berkeley National Laboratory (LBNL) led by Gregory Kurtzer.
- Singularity is emerging as the containerization framework of choice in HPC environments.
    1. **Enable researchers to package entire scientific workflows, libraries, and even data.**
    2. **Can use docker images.**
    3. **Does not require root privileges to run.**

Apptainer

A little Apptainer history

Singularity was written in C by Greg Kurtzer at LBNL in 2015, with many community contributions

Greg founded Sylabs in 2017, where singularity-3 was rewritten in Go

Greg left Sylabs in 2020, taking the Singularity project leadership with him

In May 2021, Sylabs left the project and created an open source fork, calling

Sylabs

In November 2021, the Singularity project joined the Linux Foundation, and renamed to Apptainer

Besides the name, what else changed?

The command is changed to **apptainer** from **singularity**. But the **singularity** command also works, because it is an alias for the command **apptainer**.

The variable prefixes are updated to **APPTAINER_** and **APPTAINERENV_** instead of the previous **SINGULARITY_** and **SINGULARITYENV_** (although both are accepted).

The default Apptainer installation is now unprivileged, allowing unprivileged users to build containers.

Apptainer Without Setuid

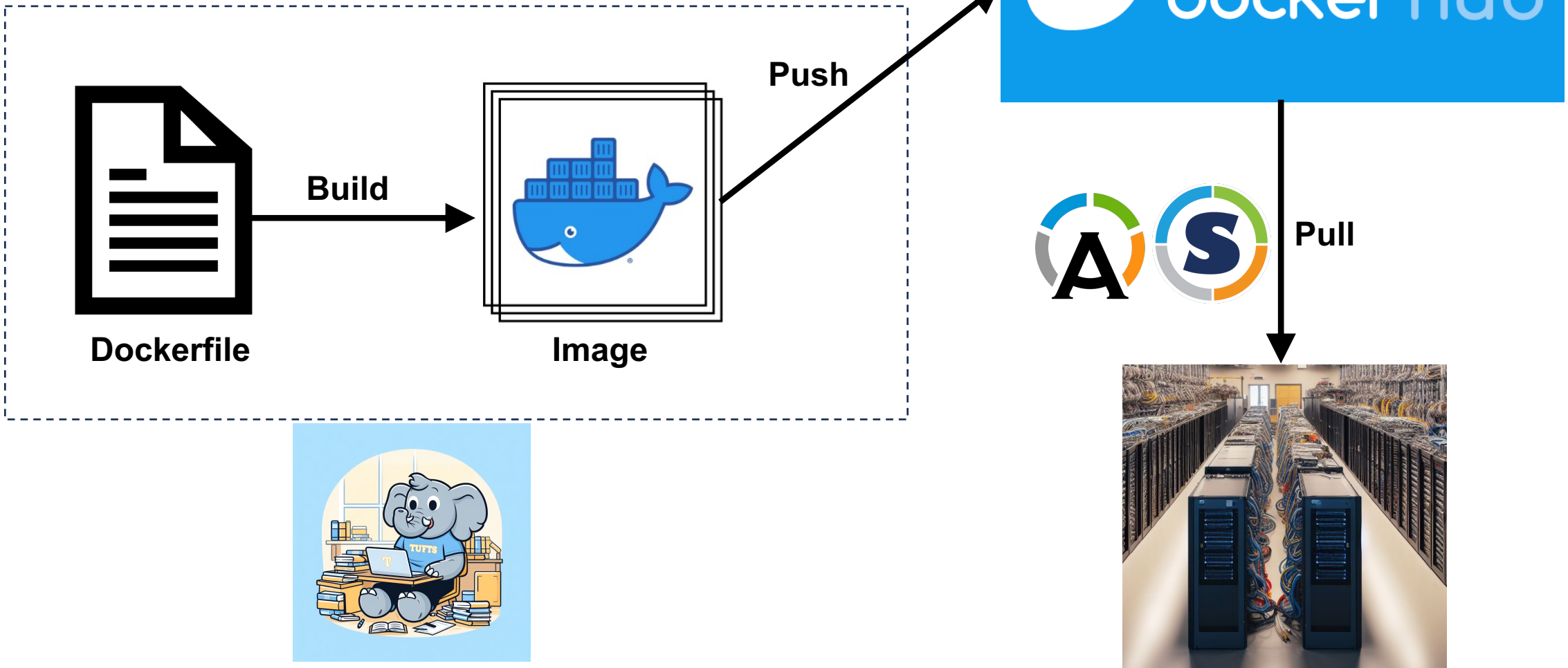
Dave Dykstra
Scientific Computing Division
Fermi National Accelerator Laboratory
Batavia, IL
https://orcid.org/0000-0003-2653-9015

***Abstract*—Apptainer (formerly known as Singularity) since its beginning implemented many of its container features with the assistance of a setuid-root program. It still supports that mode, but as of version 1.1.0 it no longer uses setuid by default. This is feasible because it now can mount squash filesystems, mount ext2/3/4 filesystems, and use overlayfs using unprivileged user namespaces and FUSE. It also now enables unprivileged users to build containers, even without requiring system administrators to configure /etc/subuid and /etc/subgid unlike other "rootless" container systems. As a result, all the unprivileged functions can be used nested inside of another container, even if the container runtime prevents any elevated privileges.**

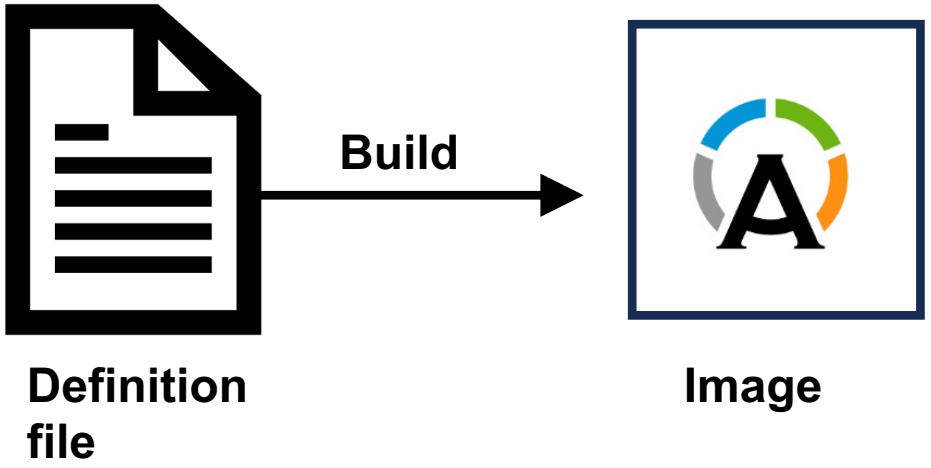
Using the default non setuid-root mode has these advantages over using the setuid-root mode:

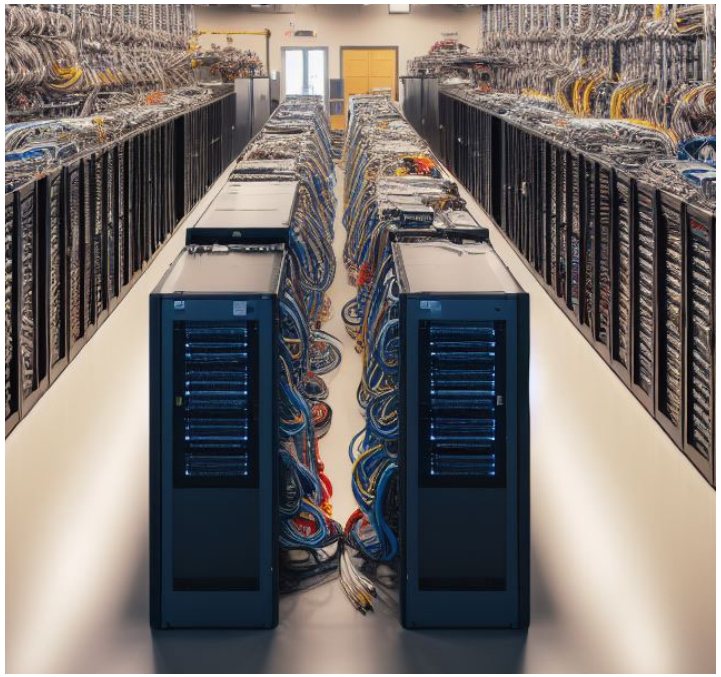
1. Setuid-root programs are notoriously difficult to make fully secure. Apptainer keeps the setuid portions to a minimum and has passed a careful review, but still it is a risk.
2. Linux kernel developers believe that it is inherently unsafe to allow unprivileged users to modify an underlying filesystem at will while kernel code is actively accessing the filesystem [5]. Kernel filesystem drivers do not and cannot protect against

How to build your own container: Docker



How to build your own container: Apptainer





Overview

- What are containers and why should we use them?
- **How to use Singularity/Apptainer to pull and run containers?**
- How to build your own containers?
- Environment modules based on containers
- How to build and run MPI and GPU containers?
- Hands-on Demo

Pulling, Running and Building Containerized HPC Applications

How to use singularity/apptainer?

Load singularity module

module purge
module load singularity/3.8.4
module list

```
[yzhang85@s1cmp007 ~]$ module purge
[yzhang85@s1cmp007 ~]$ module avail singularity

------------------------------------- /opt/shared/Modules/modulefiles-rhel6 --------
singularity/2.6.1          singularity/3.5.3          singularity/3.8.4(default)
singularity/3.1.0          singularity/3.6.1
[yzhang85@s1cmp007 ~]$ module load singularity/3.8.4
[yzhang85@s1cmp007 ~]$ module list
Currently Loaded Modulefiles:
  1) squashfs/4.4                2) singularity/3.8.4(default)
```

Singularity basics

Detailed singularity user guide is available at: **sylabs.io/guides/3.8/user-guide**

```
singularity [options] <subcommand> [subcommand options …]
```

- **pull**
- **exec**
- **shell**
- **build**
- run
- push
- Instance
- help
- …

pull: download a container from a given URI

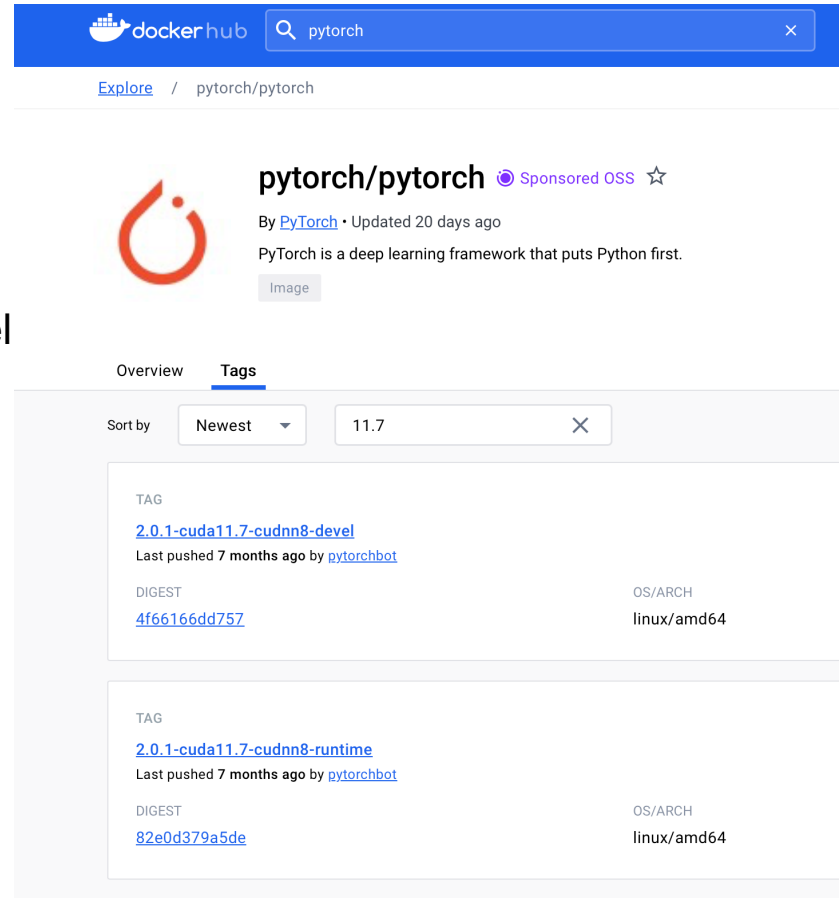
singularity pull [output file] URI

Example:

singularity pull pytorch_2.0.1.sif **docker://**pytorch/pytorch:2.0.1-cuda11.7-cudnn8-devel

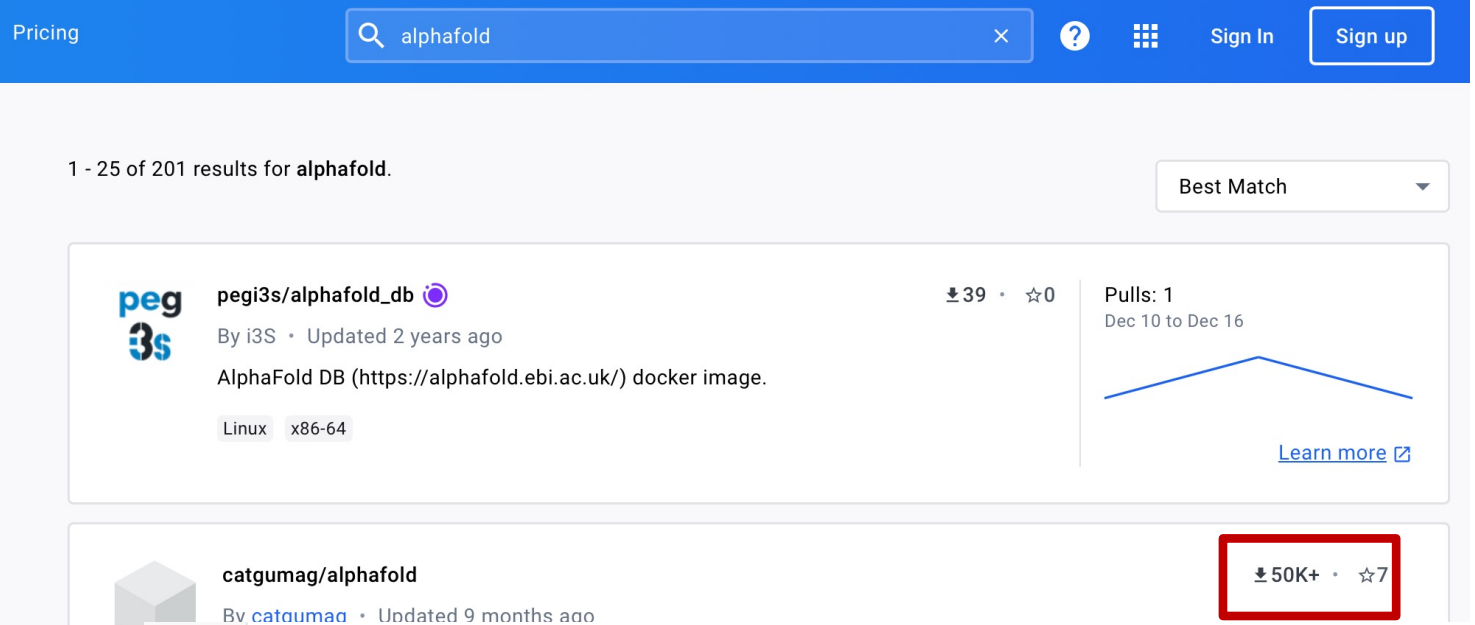
Custom Name

URI



Complete supported URIs list can be found here:

https://docs.sylabs.io/guides/3.8/user-guide/cli/singularity_pull.html



https://hub.docker.com/r/catgumag/alphafold/tags

TAG

latest

Last pushed **9 months ago** by dialvarezs

DIGEST

069598169a82

OS/ARCH

linux/amd64

COMPRESSED SIZE

2.84 GB

`docker pull catgumag/alphafold:l…`

TAG

2.3.2

Last pushed **9 months ago** by dialvarezs

DIGEST

069598169a82

OS/ARCH

linux/amd64

COMPRESSED SIZE

2.84 GB

`docker pull catgumag/alphafold:2…`

pull: examples

```
docker pull quay.io/biocontainers/blast:<tag>

(see `blast/tags`_ for valid values for ``<tag>``)
```

docker pull quay.io/biocontainers/blast:2.15.0--pl5321h6f7f691_1 → **singularity** pull **docker://**quay.io/biocontainers/blast:2.15.0--pl5321h6f7f691_1

docker pull pytorch/pytorch:2.1.2-cuda11.8-cudnn8-devel → **singularity** pull **docker://**pytorch/pytorch:2.1.2-cuda11.8-cudnn8-devel

docker pull tensorflow/tensorflow:2.15.0.post1-gpu → **singularity** pull **docker://**tensorflow/tensorflow:2.15.0.post1-gpu

shell: run a shell within a container

singularity shell [options] image

Example:

singularity shell --nv pytorch_2.1.1.sif

```
Singularity> more /etc/os-release
NAME="Ubuntu"
VERSION="20.04.5 LTS (Focal Fossa)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 20.04.5 LTS"
VERSION_ID="20.04"
```

```
Singularity> python
Python 3.10.11 (main, Apr 20 2023, 19:02:41) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> torch.cuda.is_available()
True
>>> torch.cuda.get_device_name(0)
'NVIDIA A100 80GB PCIe'
```

Singularity> exit ## To exit container, and go back to host

exec: run executables/scripts

singularity exec [options] image command

Examples:

singularity exec --nv pytorch_2.1.1.sif **python**

singularity exec –B /cluster/tufts r_4.3.1_scrnaseq.sif **Rscript** myscript.R

singularity exec trinityrnaseq_trinityrnaseq:2.15.1.sif **Trinity -h**

Bind: binding host directories into container

Programs running inside a container will not have access to directories and files outside of your home and the current directory. Singularity allows you to map directories on your host system to directories within your container using bind mounts.

Singularity binds several directories into the container image automatically.
$HOME and **/tmp** is the default list.
To enable containers to access your project directories on the cluster, you can use **-B /cluster/tufts** or **export SINGULARITY_BIND="/cluster/tufts"**

singularity shell/run/exec --bind /cluster/tufts image.sif

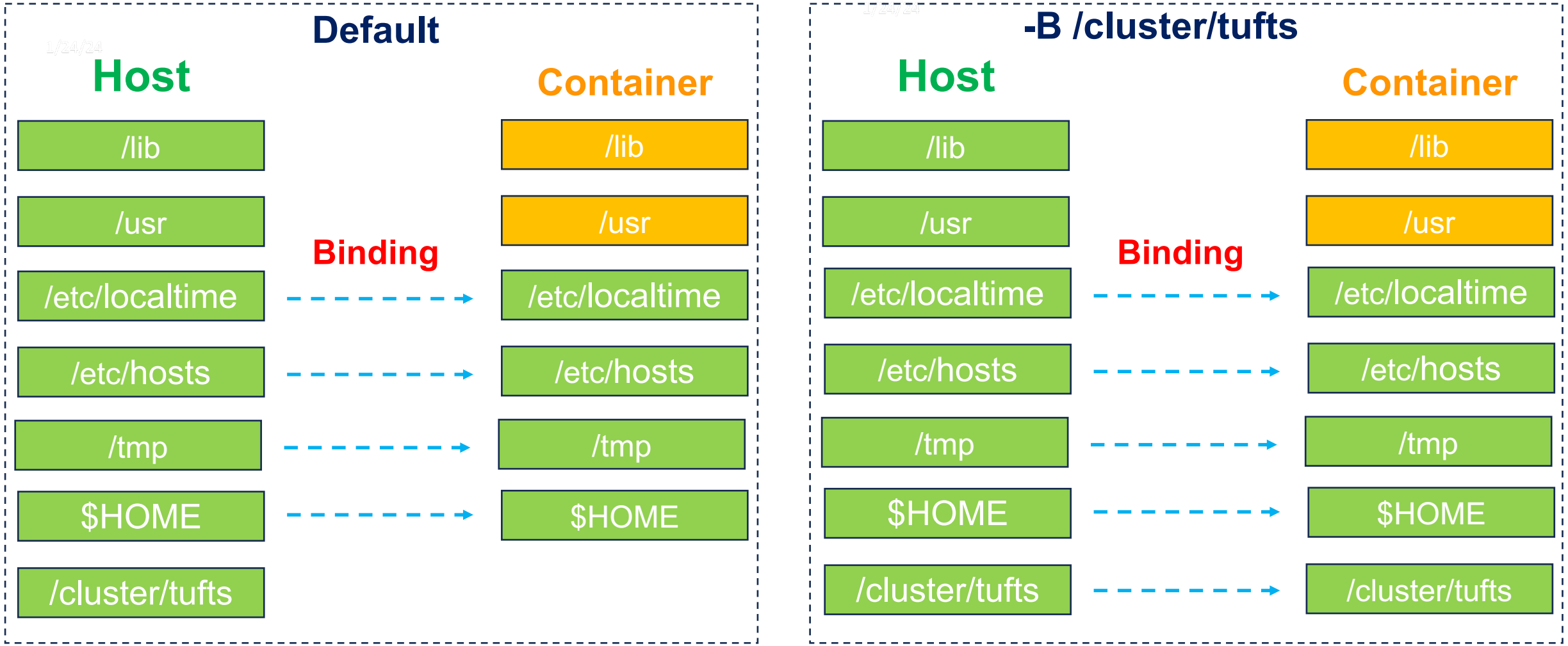
or

singularity shell/run/exec –B /cluster/tufts image.sif

or

export SINGULARITY_BIND="/cluster/tufts"

Bind host directories



Cache

$ ncdu $HOME

```
ncdu 1.15.1 ~ Use the arrow keys to navigate
--- /cluster/home/yzhang85 ----------------
   67.9 GiB [##########] /.singularity
    5.1 GiB [          ] /.apptainer
    1.3 GiB [          ] /R
  797.7 MiB [          ] /.local
  595.6 MiB [          ] /.conda
  480.3 MiB [          ] /.vscode-server
  341.7 MiB [          ] /.cache
  318.7 MiB [          ] /nf-core
  313.1 MiB [          ]  r-base_4.3.2.sif
  223.1 MiB [          ] /.config
  189.9 MiB [          ] /ood_tufs_apps
  185.4 MiB [          ] /bin
  140.4 MiB [          ] /svn
   76.2 MiB [          ] /ondemand
   74.2 MiB [          ] /.nextflow
```

To mitigate this, users can either run the singularity pull command with --disable-cache

singularity/apptainer pull --disable-cache URI

or manually clean $HOME/.singularity/cache and $HOME/.apptainer/cache

or export **SINGULARITY_CACHEDIR**=/cluster/tufts/XXXX
export **APPTAINER_CACHEDIR**=/cluster/tufts/XXXX

singularity build: build container with sudo

sudo singularity build image.sif definition.def

```
[yzhang85@s1cmp007 ~]$ singularity build openmpi_4.1.4_pmi2.sif openmpi_4.1.4_pmi2.def
FATAL:   You must be the root user, however you can use --remote or --fakeroot to build from a Singularity recipe file
[yzhang85@s1cmp007 ~]$ sudo singularity build openmpi_4.1.4_pmi2.sif openmpi_4.1.4_pmi2.def

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for yzhang85:
```

Since users do not have sudo privilege, this build method only works for computers that you have sudo.

apptainer build: build container without sudo

apptainer build is only enabled on interactive partition

```
srun -p interactive -n 2 -t 2:00:00 --pty bash
module purge
module load apptainer/1.2.5-no-suid
apptainer build image.sif definition.def
```

Overview

- What are containers and why should we use them?
- How to use Singularity/Apptainer to pull and run containers?
- **How to build your own containers?**
- Environment modules based on containers
- How to build and run MPI and GPU containers?
- Hands-on Demo

Pulling, Running and Building Containerized HPC Applications

Definition files

Definition file

A **definition** file, or **def** file, is a recipe to build a container image with singularity/apptainer. It is divided into two parts:

1. **Header**: the Header describes the core operating system to build within the container.
   - Bootstrap
   - From
2. **Section**: each section is defined by a **%** character followed by the name of the particular section. Different sections add different content or execute commands at different times during the build process.
   - help
   - setup
   - files
   - label
   - **environment**
   - **post**
   - Runscript
   - …

Header

Header references the kind of base you want to use (e.g., docker, shub).

Images hosted on Docker Hub

```
Bootstrap: docker
From: ubuntu:22.04
```

ubuntu Docker Official Image · 1B+ · 10K+

Ubuntu is a Debian-based Linux operating system based on free software.

```
Bootstrap: docker
From: nvidia/cuda:12.1.0-cudnn8-devel-ubuntu22.04
```

nvidia/cuda ☆

By nvidia • Updated 6 days ago

CUDA and cuDNN images from gitlab.com/nvidia/cuda

Image

Images saved on your machine

```
Bootstrap: localimage
From: /cluster/tufts/biocontainers/base_images/mamba_1.4.2.sif
```

Section: %label and %help

The **label**s section allows you to define metadata for your container.

The **help** section shows the user-defined help for an image.

```
%labels
Author "Yucheng Zhang <yzhang85@tufts.edu>“
Version 4.3.2

%help
This is an Ubuntu 20.04 container with R/4.3.2 and Rstudio/2023.06-2
```

```
$ singularity inspect  --labels <image path>
$ singularity run-help <image path>
```

Section: %files and %environment

To copy files from the host to the container, use the **%files** command. Each line in **%files** should have a source and destination path.

The source path should be on your host, while the destination path should be a location within the container.

%environment allows you to define environment variables for your container. These variables are available when you run the container, not during its build.

```
%files
        install.sh          /opt/install.sh
        example_data/       /opt/example_data
%environment
        export PATH=/opt/myapp/bin:$PATH
        export LD_LIBRARY_PATH=/opt/myapp/lib:$LD_LIBRARY_PATH
        export LC_ALL=C
```

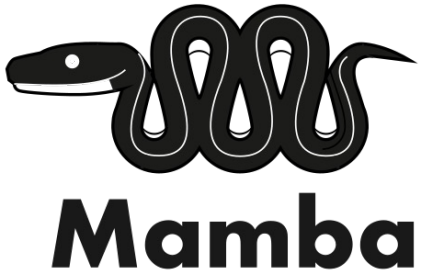
Section: %post

%post is where you can download files, install new software and libraries, write configuration files, and create directories using tools like git and wget.

```
Bootstrap: docker
From: ubuntu:18.04

%post
# Update and install system libraries
apt-get update
apt-get -y install --no-install-recommends --no-install-suggests build-essential libssl-dev wget

# KALIGN 2.0.4
cd /opt  && mkdir kalign2
cd kalign2 && wget http://msa.sbc.su.se/downloads/kalign/current.tar.gz
tar -xvf current.tar.gz && ./configure && make

%environment
export PATH=/opt/kalign2:$PATH
```

Bootstrap: localimage

```
Bootstrap: localimage
From: /cluster/tufts/biocontainers/base_images/mamba_1.4.2.sif

%labels
        Author "Yucheng Zhang <yzhang85@tufts.edu>"

%post
        mamba install -c bioconda fastqc trim-galore samtools \
            bowtie2 star subread hisat2 salmon \
            picard bedtools stringtie
```



Mamba can speed up your conda installs by 50-80%.

Overview

- What are containers and why should we use them?
- How to use Singularity/Apptainer to pull and run containers?
- How to build your own containers?
- **Environment modules based on containers**
- How to build and run MPI and GPU containers?
- Hands-on Demo

Pulling, Running and Building Containerized HPC Applications

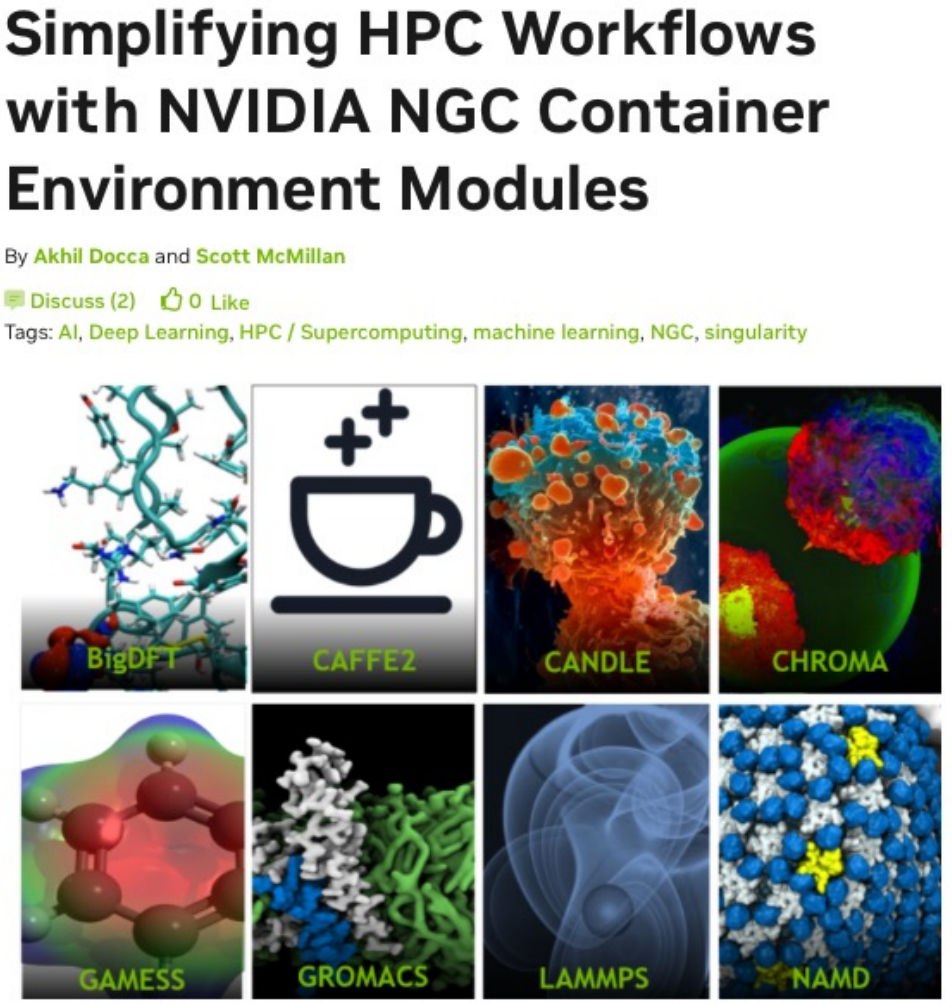
Environment modules

NGC container environment modules

NGC container environment modules are lightweight wrappers that make it possible to transparently use NGC containers as environment modules.

1. Allow HPC users to utilize familiar environment module commands.
2. Leverage all the benefits of containers, including portability and reproducibility.

https://github.com/NVIDIA/ngc-container-environment-modules



catalog.ngc.nvidia.com

NVIDIA NGC | CATALOG

Welcome Guest

All You Need to Build AI. All in One Place.

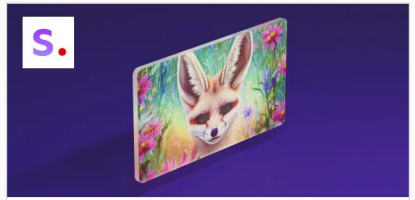
Welcome to the NGC Catalog - GPU Accelerated AI models and SDKs that help you infuse AI into your applications at speed of light

Explore Use Cases

NVIDIA NGC: AI Development Catalog

Search NGC Catalog...

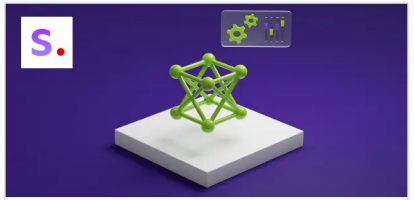
NVIDIA AI Foundation Models



Stable Video Diffusion

Stable Video Diffusion (SVD) Image-to-Video is a diffusion model that takes in a still image as a conditioning frame, and...

View Labels | Learn More



SDXL Turbo

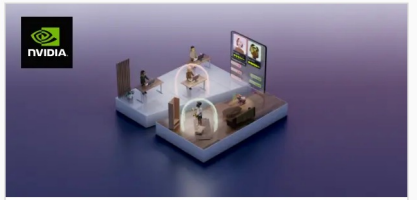
SDXL-Turbo is a fast generative text-to-image model that can synthesize photorealistic images from a text promp...

View Labels | Learn More

Llama Guard

Llama Guard is a model that provides input and output guardrails for LLM deployments.

View Labels | Learn More



Maxine Voice Font

Voice Font converts the speaker's voice in input audio to match the target voice while keeping linguistic information and...

View Labels | Learn More

View More

Nvidia GPU-optimized tools for deep learning, machine learning, and high-performance computing.

https://catalog.ngc.nvidia.com/

```
[yzhang85@login-prod-01 ~]$ module load ngc
[yzhang85@login-prod-01 ~]$ module avail

------------------------------------------------ /cluster/tufts/ngc/modules -------------------------------
autodock/2020.06            lammps/15Jun2023            pytorch/21.06              tensorflow/21.06-tf2
chroma/2020.06              milc/quda0.8-patch4Oct2017  pytorch/21.09              tensorflow/21.09-tf1
chroma/2021.04              namd/3.0-alpha3-singlenode  pytorch/2.1.2              tensorflow/21.09-tf2
cuda-quantum/0.4.0          namd/3.0-beta5              pytorch/23.12              tensorflow/2.15.0
cuquantum-appliance/23.03   nvhpc/20.11                 qmcpack/3.5.0              tensorflow/23.12-tf2
gamess/17.09-r2-libcchem    nvhpc/20.7                  quantum_espresso/7.1       torchani/2021.04
gromacs/2021.3              nvhpc/20.9                  rapidsai/21.06
gromacs/2023.2              nvhpc/21.5                  rapidsai/21.10
lammps/10Feb2021            nvhpc/21.9                  tensorflow/21.06-tf1
```

 # Load ngc

module load ngc

 # Check available applications

module avail

 # Load and run specific tools

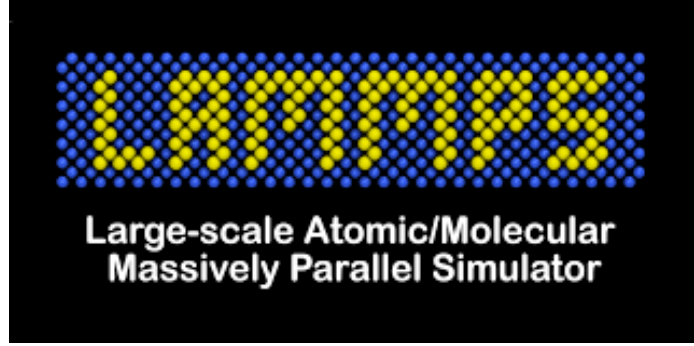
module load pytorch/21.09

No need to load cuda and cudnn modules

```
[yzhang85@s1cmp007 ~]$ module load ngc
[yzhang85@s1cmp007 ~]$ module load pytorch/21.09
[yzhang85@s1cmp007 ~]$ python
BIOC: Enabling Nvidia GPU support in the container.
Python 3.8.10 | packaged by conda-forge | (default, May 11 2021, 07:01:05)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> torch.cuda.device_count()
1
>>> torch.cuda.get_device_name(0)
'NVIDIA A100 80GB PCIe'
>>> a = torch.Tensor([[1,2],[3,4]])
>>> print(a)
tensor([[1., 2.],
        [3., 4.]])
>>> print(a**2)
tensor([[ 1.,  4.],
        [ 9., 16.]])
>>>
```

Note: to use NGC containers, you also need to use GPU nodes equipped with Nvidia GPUs.

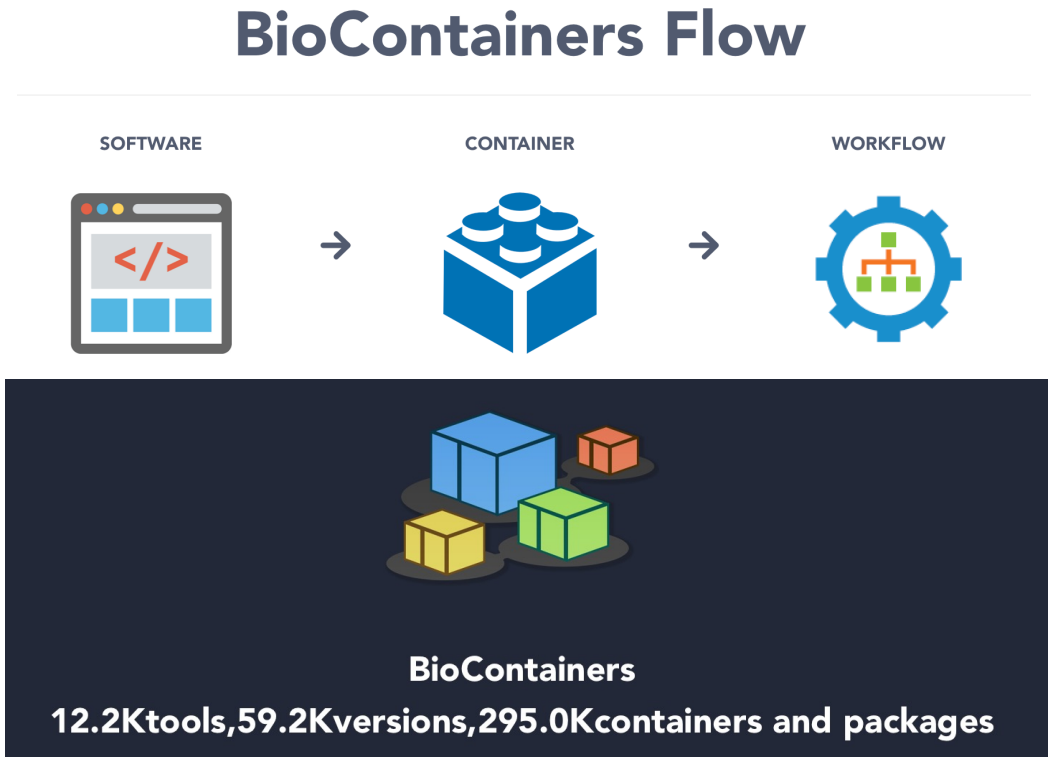
```
[yzhang85@login-prod-01 lammps]$ module purge
[yzhang85@login-prod-01 lammps]$ module load ngc
[yzhang85@login-prod-01 lammps]$ module load lammps/15Jun2023
[yzhang85@login-prod-01 lammps]$ srun --mpi=pmi2 -p gpu -N1 -n4 -c2 --gres=gpu:2 lmp -k on g 2 -sf kk -pk kokk
os cuda/aware on neigh full comm device binsize 2.8 -var x 8 -var y 8 -var z 8  -in in.lj
srun: job 202027 queued and waiting for resources
srun: job 202027 has been allocated resources
BIOC: Enabling Nvidia GPU support in the container.
BIOC: Enabling Nvidia GPU support in the container.
BIOC: Enabling Nvidia GPU support in the container.
BIOC: Enabling Nvidia GPU support in the container.
level=info msg="Setting CUDA_MODULE_LOADING=LAZY. Please see https://docs.nvidia.com for more information."
LAMMPS (15 Jun 2023)
KOKKOS mode is enabled (src/KOKKOS/kokkos.cpp:107)
  will use up to 2 GPU(s) per node
  using 1 OpenMP thread(s) per MPI task
Lattice spacing in x,y,z = 1.6795962 1.6795962 1.6795962
Created orthogonal box = (0 0 0) to (268.73539 268.73539 268.73539)
  1 by 2 by 2 MPI processor grid
Created 16384000 atoms
  using lattice units in orthogonal box = (0 0 0) to (268.73539 268.73539 268.73539)
  create_atoms CPU = 0.743 seconds
Generated 0 of 0 mixed pair_coeff terms from geometric mixing rule
Neighbor list info ...
  update: every = 20 steps, delay = 0 steps, check = no
  max neighbors/atom: 2000, page size: 100000
  master list distance cutoff = 2.8
  ghost atom cutoff = 2.8
  binsize = 2.8, bins = 96 96 96
  1 neighbor lists, perpetual/occasional/extra = 1 0 0
  (1) pair lj/cut/kk, perpetual
      attributes: full, newton off, kokkos_device
      pair build: full/bin/kk/device
      stencil: full/bin/3d
      bin: kk/device
Setting up Verlet run ...
  Unit style    : lj
  Current step  : 0
  Time step     : 0.005
Per MPI rank memory allocation (min/avg/max) = 578 | 578 | 578 Mbytes
   Step          Temp          E_pair         E_mol          TotEng         Press
         0   1.44          -6.7733681      0             -4.6133682     -5.0196693
       100   0.75918554    -5.7610896      0             -4.6223114      0.19195085
```



BioContainers

BIOCONDA®

- BioContainers is integrated with Bioconda, which is the conda channel for bioinformatics applications.
- BioContainers registry is the largest registry for bioinformatics applications.
- As of today, BioContainers provides containers for over 12 thousand bioinformatics applications.



You can find almost all bioinformatics applications from here:
https://bioconda.github.io/conda-package_index.html

Biocontainers @ Tufts

```
------------------------------------------ /cluster/tufts/biocontainers/modules ------------------------------
alphafold/2.3.0                dorado/0.5.0                minipolish/0.1.3            r-scrnaseq/4.3.1
alphafold/2.3.1                exonerate/2.4.0             mirdeep2/2.0.1.3            salmon/1.10.1
alphafold/2.3.2                fasta3/36.3.8               mirge3/0.1.4                salmon/1.9.0
angsd/0.939                    fastp/0.23.2                mothur/1.46.0               samtools/1.16
angsd/0.940                    fastqc/0.12.1               mothur/1.47.0               samtools/1.17
bcftools/1.13                  fastspar/1.0.0              mothur/1.48.0               signalp6/6.0-fast
bcftools/1.14                  filtlong/0.2.1              nextflow/23.10.0            signalp6/6.0-slow
bcftools/1.17                  flye/2.9                    nf-core/2.10                spades/3.15.4
beast2/2.6.3                   flye/2.9.1                  ngc/1.0                     spades/3.15.5
beast2/2.6.4                   flye/2.9.2                  orthofinder/2.5.5           squid/1.5
beast2/2.6.6                   gatk4/4.2.6.1               pandaseq/2.11               star/2.7.10a
bedops/2.4.39                  gatk4/4.3.0.0               parabricks/4.0.0-1          star/2.7.10b
bedtools/2.30.0                guppy/6.4.6                 parabricks/4.2.1-1          star/2.7.11a
bedtools/2.31.0                guppy/6.5.7                 pepper_deepvariant/r0.8     star/2.7.9a
biobakery_workflows/3.0.0.a.7  hap.py/0.3.12              pepper_deepvariant/r0.8-gpu subread/1.6.4
blast/2.13.0                   hisat2/2.2.1                picard/2.25.1               subread/2.0.1
bowtie2/2.4.2                  hmmer/3.3.2                 picard/2.26.10              tmhmm/2.0c
bowtie2/2.5.1                  homer/4.11                  plink/1.90b6.21             transdecoder/5.5.0
breseq/0.38.2                  htseq/2.0.2                 plink2/2.00a2.3             transdecoder/5.7.1
busco/5.4.1                    humann/3.8                  polypolish/0.5.0            trim-galore/0.6.10
busco/5.4.7                    hyper-shell/2.4.0           qiime2/2023.2               trim-galore/0.6.7
canu/2.1.1                     impute2/2.3.2               qiime2/2023.5               trimmomatic/0.39
canu/2.2                       kallisto/0.46.2             qiime2/2023.7               trinity/2.15.0
cellprofiler/4.2.6             kallisto/0.48.0             qiime2/2023.9               trinity/2.15.1
cellranger/7.0.1               macs2/2.2.7.1               qualimap/2.2.1              trinotate/4.0.2
cellranger/7.1.0               macs3/3.0.0a6               qualimap/2.3                trycycler/0.5.0
cellranger-atac/2.0.0          masurca/4.0.9               raven-assembler/1.8.1       trycycler/0.5.3
cellranger-atac/2.1.0          masurca/4.1.0               raxml-ng-mpi/1.2.0          trycycler/0.5.4
cufflinks/2.2.1                medaka/1.11.1               relion/4.0.1                vcftools/0.1.16
cutadapt/3.4                   megahit/1.2.9               rmats2sashimiplot/2.0.4     viennarna/2.5.0
cutadapt/3.7                   metaphlan/4.0.2             rmats2sashimiplot/3.0.0
diamond/2.0.15                 miniasm/0.3_r179            rnaquast/2.2.3
diamond/2.1.6                  minimap2/2.26               r-scrnaseq/4.2.3
```

TACC
TEXAS ADVANCED COMPUTING CENTER

PSC

PURDUE UNIVERSITY

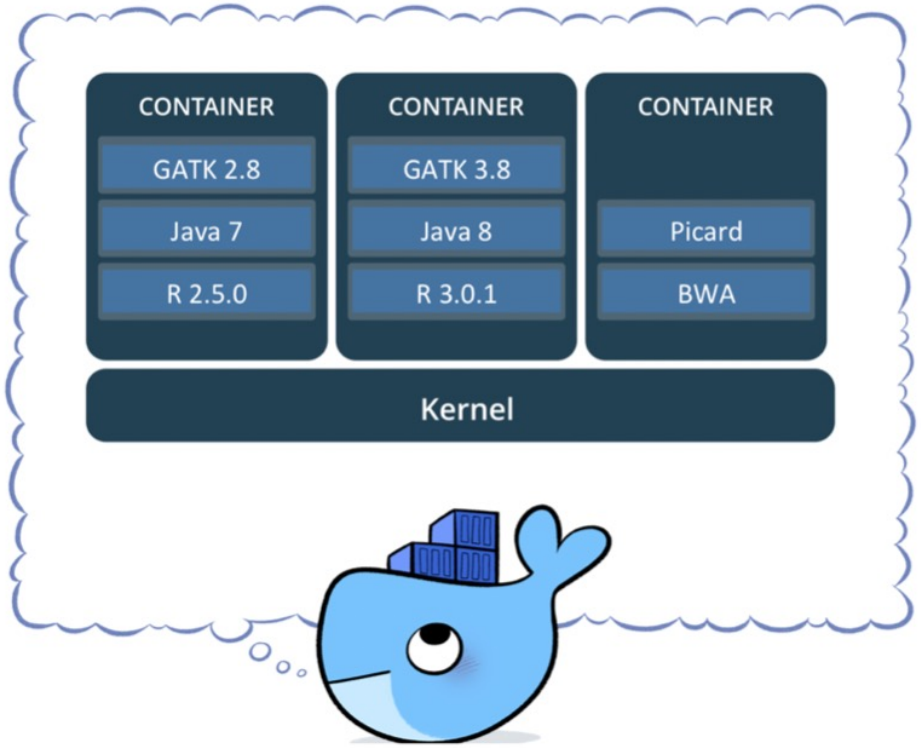
Rosen Center for Advanced Computing

At Tufts HPC, we look forward to deploying bioinformatics applications based on containers wherever possible.

Only load the module for target application

```
module load gatk/2.8
```

```
module load java/7
module load r/2.5.0
module load gatk/2.8
```

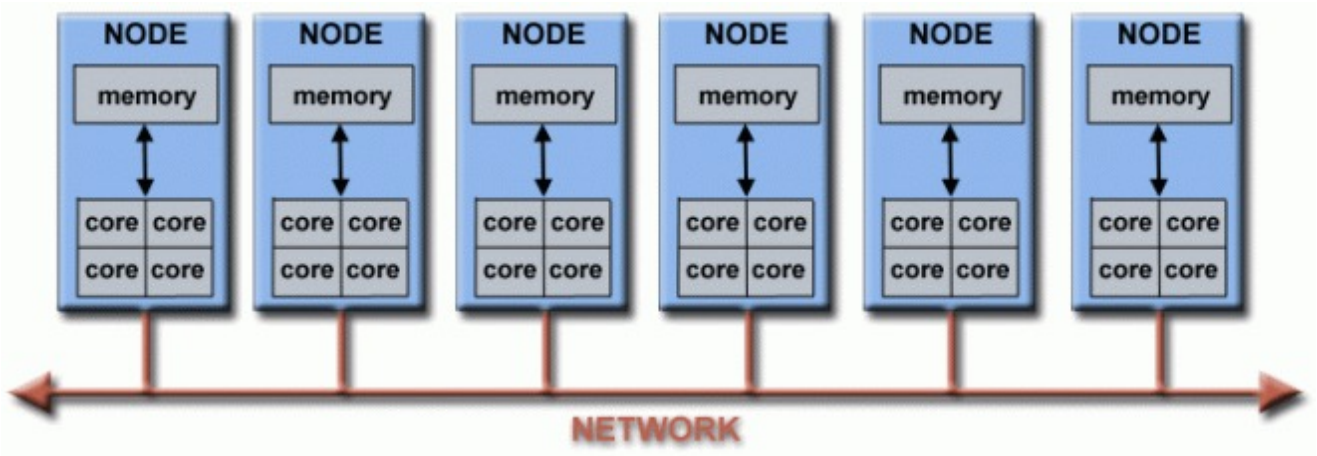


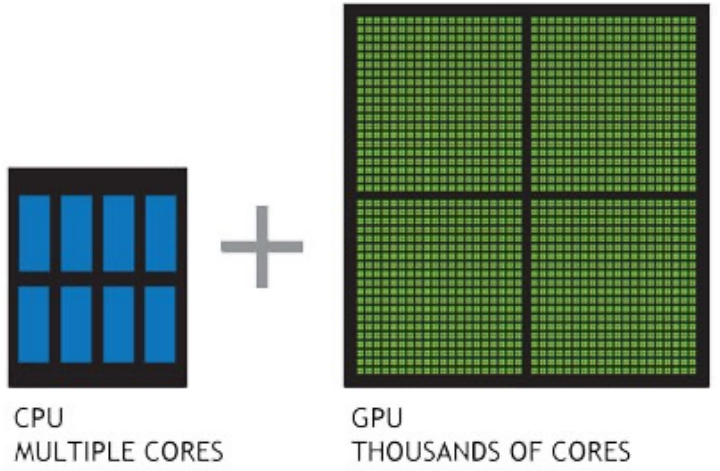
Overview

- What are containers and why should we use them?
- How to use Singularity/Apptainer to pull and run containers?
- How to build your own containers?
- Environment modules based on containers
- **How to build and run MPI and GPU containers?**
- Hands-on Demo

Pulling, Running and Building Containerized HPC Applications

Advanced topics: MPI and GPU





MPI (Message Passing Interface)

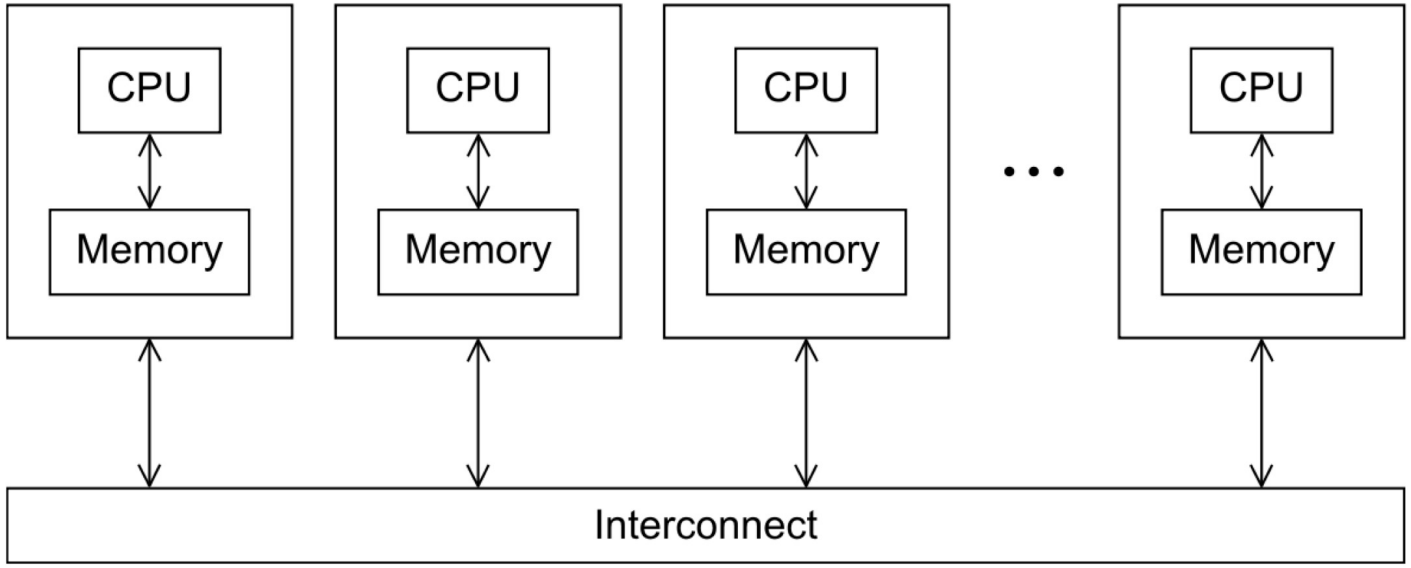
Although singularity/apptainer has always supported MPI-enabled applications, it can be challenging to get them working across multiple nodes.

Two old ways (https://apptainer.org/docs/user/1.0/mpi.html)

1. Hybrid model
2. bind model

The new way

- PMI



An Introduction to Parallel Programming by Peter Pacheco and Matthew Malensek

MPI: hybrid model

- This model is called "**hybrid**" because it requires both an MPI implementation on the host and another implementation in the container image.
- The host MPI provides the **mpirun** or **mpiexec** command and start ranks or containers on computing nodes.
- MPI in the container image is used to run the application.
- Host and container MPI need to be "**compatible**" since they need to tightly interact with each other.

```
module load openmpi/XXX ## load the MPI module compatible with your container
module load singularity/3.8.4
mpirun -n $SLURM_NTASKS singularity exec mpi_container.sif mpi_program
```

MPI: bind model

- The main difference between the hybrid and bind mode is the fact that with the bind mode, the container usually does not include any MPI implementation.
- This means that **singularity/apptainer needs to mount/bind the MPI and high-speed interconnect libraries available on the host into the container**.

Technically this requires two steps:

1. Know where the MPI implementation on the host is installed.
2. Mount/bind it into the container in a location where the system will be able to find libraries and binaries.

Binding the complete paths is difficult and inconvenient

```
md@zeus-1:openfoam$ module show singularity
-------------------------------------------------------------------------------------------------------------------
   /pawsey/sles12sp3/modulefiles/devel/singularity/3.5.2.lua:
-------------------------------------------------------------------------------------------------------------------
help([[Sets up the paths you need to use singularity version 3.5.2]])
whatis("Singularity enables users to have full control of their environment. Singularity
containers can be used to package entire scientific workflows, software and
libraries, and even data.

For further information see https://sylabs.io/singularity")
whatis("Compiled with gcc/4.8.5")
setenv("MAALI_SINGULARITY_HOME","/pawsey/sles12sp3/devel/gcc/4.8.5/singularity/3.5.2")
prepend_path("MANPATH","/pawsey/sles12sp3/devel/gcc/4.8.5/singularity/3.5.2/share/man")
prepend_path("PATH","/pawsey/sles12sp3/devel/gcc/4.8.5/singularity/3.5.2/bin")
setenv("SINGULARITYENV_LD_LIBRARY_PATH","/usr/lib64:/pawsey/intel/17.0.5/compilers_and_libraries/linux/mpi/intel64/lib"
)
setenv("SINGULARITY_BINDPATH","/astro,/group,/scratch,/pawsey,/etc/dat.conf,/etc/libibverbs.d,/usr/lib64/libdaplofa.so.
2,/usr/lib64/libdaplofa.so.2.0.0,/usr/lib64/libdat2.so.2,/usr/lib64/libdat2.so.2.0.0,/usr/lib64/libibverbs,/usr/lib64/l
ibibverbs.so,/usr/lib64/libibverbs.so.1,/usr/lib64/libibverbs.so.1.1.14,/usr/lib64/libmlx5.so,/usr/lib64/libmlx5.so.1,/
usr/lib64/libmlx5.so.1.1.14,/usr/lib64/libnl-3.so.200,/usr/lib64/libnl-3.so.200.18.0,/usr/lib64/libnl-cli-3.so.200,/usr
/lib64/libnl-cli-3.so.200.18.0,/usr/lib64/libnl-genl-3.so.200,/usr/lib64/libnl-genl-3.so.200.18.0,/usr/lib64/libnl-idia
g-3.so.200,/usr/lib64/libnl-idiag-3.so.200.18.0,/usr/lib64/libnl-nf-3.so.200,/usr/lib64/libnl-nf-3.so.200.18.0,/usr/lib
64/libnl-route-3.so.200,/usr/lib64/libnl-route-3.so.200.18.0,/usr/lib64/librdmacm.so,/usr/lib64/librdmacm.so.1,/usr/lib
64/librdmacm.so.1.0.14")
setenv("SINGULARITY_CACHEDIR","/group/pawsey0001/mdelapierre/.singularity")
```

https://www.youtube.com/watch?v=sV4JmbWUirc&t=4s

MPI: srun and PMI(Process Manager Interface)

The basic idea behind this approach is to put the entire MPI framework into a container along with the MPI application and then to use a tool (e.g. slurm) that implements one of the PMI standards to launch the MPI jobs.

MPI inside container need to be built with the same PMI standard support.
On Tufts cluster, **PMI-2 is recommended.**

Home / CIQ Blog / Apptainer / Singularity

A New Approach to MPI in Apptainer

Dave Godlove • June 27, 2023

https://ciq.com/blog/a-new-approach-to-mpi-in-apptainer

MPI: srun and PMI

module purge # mainly purpose is to unload host mpi modules
module load singularity
srun --mpi=**pmi2** singularity exec mpi_image.sif mpi_application

```
[yzhang85@login-prod-02 ~]$ module purge
[yzhang85@login-prod-02 ~]$ module load singularity/3.8.4
[yzhang85@login-prod-02 ~]$ srun --mpi=pmi2 -N4 -n8 --ntasks-per-node=2 singularity exec openmpi_4.1.5_pmi2.sif /opt/mpi_hello_world
Hello world! Processor p1cmp045.pax.tufts.edu, Rank 6 of 8, CPU 0, NUMA node 0, Namespace mnt:[4026533512]
Hello world! Processor p1cmp045.pax.tufts.edu, Rank 7 of 8, CPU 40, NUMA node 0, Namespace mnt:[4026533513]
Hello world! Processor p1cmp043.pax.tufts.edu, Rank 2 of 8, CPU 40, NUMA node 0, Namespace mnt:[4026533514]
Hello world! Processor p1cmp043.pax.tufts.edu, Rank 3 of 8, CPU 0, NUMA node 0, Namespace mnt:[4026533513]
Hello world! Processor p1cmp044.pax.tufts.edu, Rank 4 of 8, CPU 0, NUMA node 0, Namespace mnt:[4026533512]
Hello world! Processor p1cmp044.pax.tufts.edu, Rank 5 of 8, CPU 40, NUMA node 0, Namespace mnt:[4026533513]
Hello world! Processor p1cmp028.pax.tufts.edu, Rank 0 of 8, CPU 0, NUMA node 0, Namespace mnt:[4026533514]
Hello world! Processor p1cmp028.pax.tufts.edu, Rank 1 of 8, CPU 0, NUMA node 0, Namespace mnt:[4026533513]
```

Tufts UNIVERSITY

```
zhan4429@scholar-fe01:~ $ module --force purge
zhan4429@scholar-fe01:~ $ srun --mpi=pmi2 -A scholar -N4 -n8 --ntasks-per-node=2 singularity -s exec openmpi_4.1.5_pmi2.sif /opt/mpi_hello_world
Hello world! Processor scholar-a001.rcac.purdue.edu, Rank 1 of 8, CPU 48, NUMA node 3, Namespace mnt:[4026534985]
Hello world! Processor scholar-a001.rcac.purdue.edu, Rank 0 of 8, CPU 13, NUMA node 0, Namespace mnt:[4026534986]
Hello world! Processor scholar-a002.rcac.purdue.edu, Rank 3 of 8, CPU 16, NUMA node 1, Namespace mnt:[4026534969]
Hello world! Processor scholar-a002.rcac.purdue.edu, Rank 2 of 8, CPU 0, NUMA node 0, Namespace mnt:[4026534970]
Hello world! Processor scholar-a003.rcac.purdue.edu, Rank 4 of 8, CPU 0, NUMA node 0, Namespace mnt:[4026534974]
Hello world! Processor scholar-a003.rcac.purdue.edu, Rank 5 of 8, CPU 16, NUMA node 1, Namespace mnt:[4026534973]
Hello world! Processor scholar-b000.rcac.purdue.edu, Rank 7 of 8, CPU 32, NUMA node 2, Namespace mnt:[4026533375]
Hello world! Processor scholar-b000.rcac.purdue.edu, Rank 6 of 8, CPU 5, NUMA node 0, Namespace mnt:[4026533376]
```

PURDUE UNIVERSITY

Building your own MPI containers

raxml-ng-mpi_1.2.0.def

```
Bootstrap: localimage
From:
/cluster/tufts/biocontainers/base_images/openmpi_4.1.5_pmi2.sif

%post
## Install dependencies
yum -y update
yum -y install flex bison gmp-devel
## Download and install raxml-ng
cd /opt
git clone --recursive https://github.com/amkozlov/raxml-ng
cd raxml-ng
mkdir build && cd build
cmake -DUSE_MPI=ON ..
make
make install
```

```
apptainer build raxml-ng-mpi_1.2.0.sif raxml-ng-mpi_1.2.0.def
```

```
#!/bin/bash
#SBATCH -p mpi
#SBATCH -N 2
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=20
#SBATCH -t 12:00:00
#SBATCH --job-name=raxml-ng
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --mail-user=XXX@tufts.edu
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module purge
module load singularity/3.8.4
export SINGULARITY_BINDPATH=/cluster/tufts
srun --mpi=pmi2 singularity exec \
        raxml-ng-mpi_1.2.0.sif raxml-ng-mpi --msa input.fas \
        --model GTR+G --threads 20 --bs-trees 1000
```

Nvidia GPU

For many applications, CPU compute resources provide sufficient performance. However, for a certain class of applications, the massively parallel compute power offered by GPUs can speed up operations by orders of magnitude.

Run a container with Nvidia GPU acceleration

singularity shell/run/exec **--nv** myimage.sif [command] [argument]

There is no need to load cuda and cudnn modules.

```
[yzhang85@s1cmp007 ~]$ nvidia-smi
Tue Dec 12 10:32:19 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 515.65.01    Driver Version: 515.65.01    CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA A100 80G...  On   | 00000000:3D:00.0 Off |                    0 |
| N/A   32C    P0    52W / 300W |      0MiB / 81920MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
```

```
[yzhang85@s1cmp007 ~]$ nvidia-smi
Sat Jan 13 15:43:48 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.129.03             Driver Version: 535.129.03   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA A100 80GB PCIe          On  | 00000000:3D:00.0 Off |                    0 |
| N/A   38C    P0              76W / 300W |   1211MiB / 81920MiB |     22%      Default |
|                                         |                      |             Disabled |
+-----------------------------------------+----------------------+----------------------+
|   1  NVIDIA A100 80GB PCIe          On  | 00000000:3F:00.0 Off |                    0 |
| N/A   40C    P0              75W / 300W |   1013MiB / 81920MiB |     22%      Default |
|                                         |                      |             Disabled |
+-----------------------------------------+----------------------+----------------------+
```

Use *-devel-* if you need to compile an application with cuda support

docker hub Explore Pricing nvidia/cuda

Explore / nvidia/cuda

nvidia/cuda ☆

By nvidia • Updated 7 days ago

CUDA and cuDNN images from gitlab.com/nvidia/cuda

Image

Overview Tags

Sort by Newest 11.7

TAG

11.7.1-devel-ubi8

Last pushed a month ago by svccomputepackagin363

DIGEST	OS/ARCH
b95d2011853f	linux/amd64
6bd5a7e0d1f8	linux/arm64
d5a326dfea8a	linux/ppc64le

TAG

11.7.1-runtime-ubi8

Last pushed a month ago by svccomputepackagin363

DIGEST	OS/ARCH
01882cd75b9f	linux/amd64
06e8d0caacd2	linux/arm64
91e872346177	linux/ppc64le

```
Bootstrap: docker
From: nvidia/cuda:11.8.0-cudnn8-devel-ubuntu22.04

%post
apt-get update && apt-get install -y \
    build-essential \
    python3 \
    python3-dev \
    python3-pip

# Install torch
pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118

# Install tensorflow
pip3 install tensorflow[and-cuda]
```

nvidia/cuda:11.8.0-cudnn8-devel-ubuntu22.04

DIGEST: sha256:bd746eb3b9953805ebe644847a227e218b5da775f47007c69930569a75c9ad7d

OS/ARCH	COMPRESSED SIZE	LAST PUSHED
linux/amd64	5 GB	2 months ago by svccomputepackagin363

TensorFlow

PyTorch

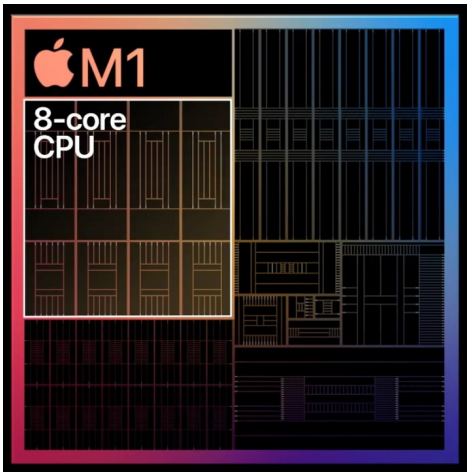
```
apptainer build pytorch2.1.2_tf2.15.0.sif pytorch2.1.2_tf2.15.0.def
singularity exec --nv pytorch2.1.2_tf2.15.0.sif  python3
```

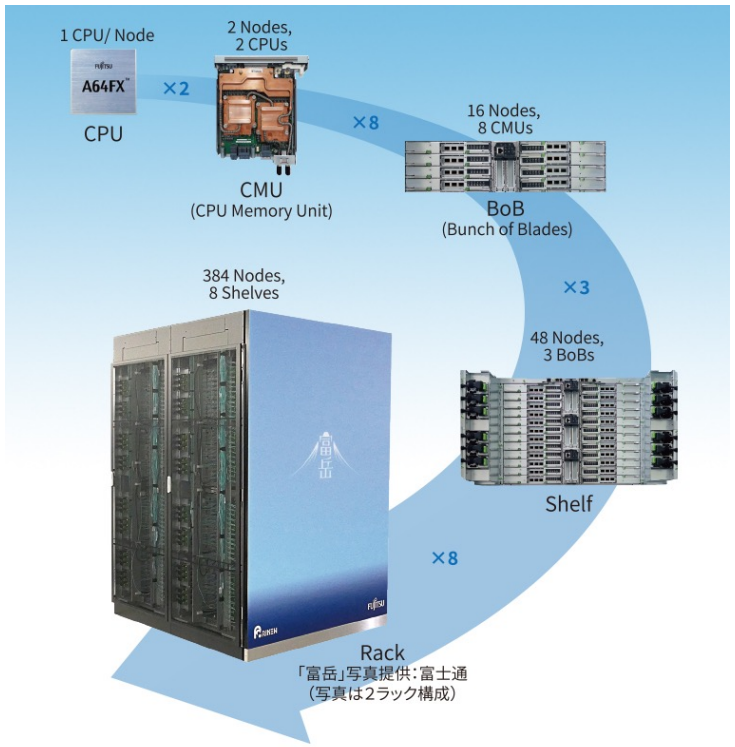
Summary

- Always use **singularity/3.8.4** to pull and run containers
- Use **apptainer/1.2.5-no-suid** only when building containers on the **interactive** partition.
- Please ensure that you bind **/cluster/tufts** to the container.
- To ensure that you're not wasting your time building your own containers, it's recommended to check if there are any publicly available containers that can serve your purpose.
- If you plan to use GPU containers, ensure compatibility between the CUDA version that was used to build the target application inside container and our GPU's CUDA driver version.
- When running containers that require GPU, make sure to include the **--nv** flag.
- Remember to use **ncdu**, a helpful tool, to regularly check and clean up **$HOME/.apptainer** and $**HOME/.singularity** directories.

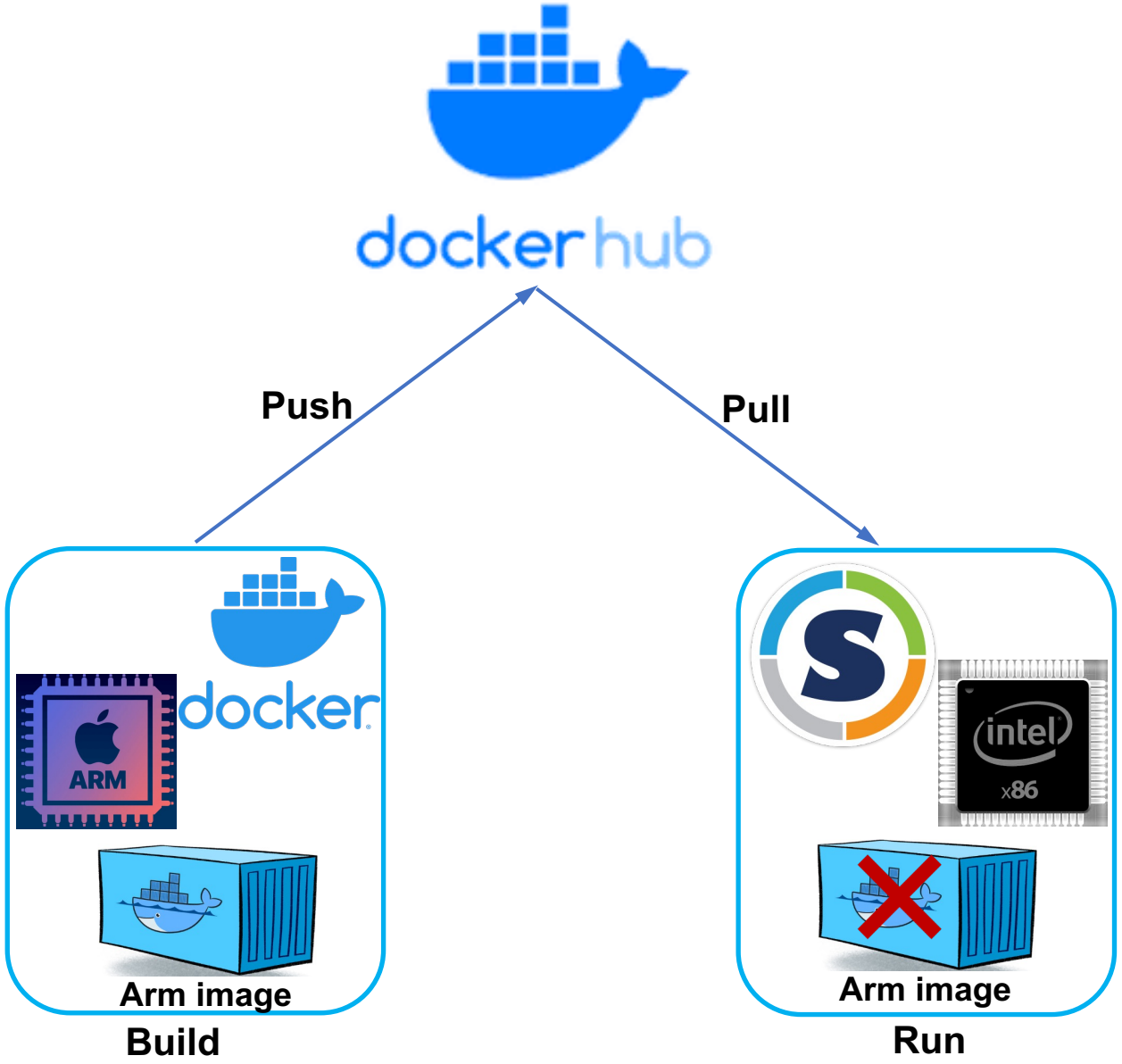
Pulling, Running and Building Containerized HPC Applications

One more thing: docker build with ARM

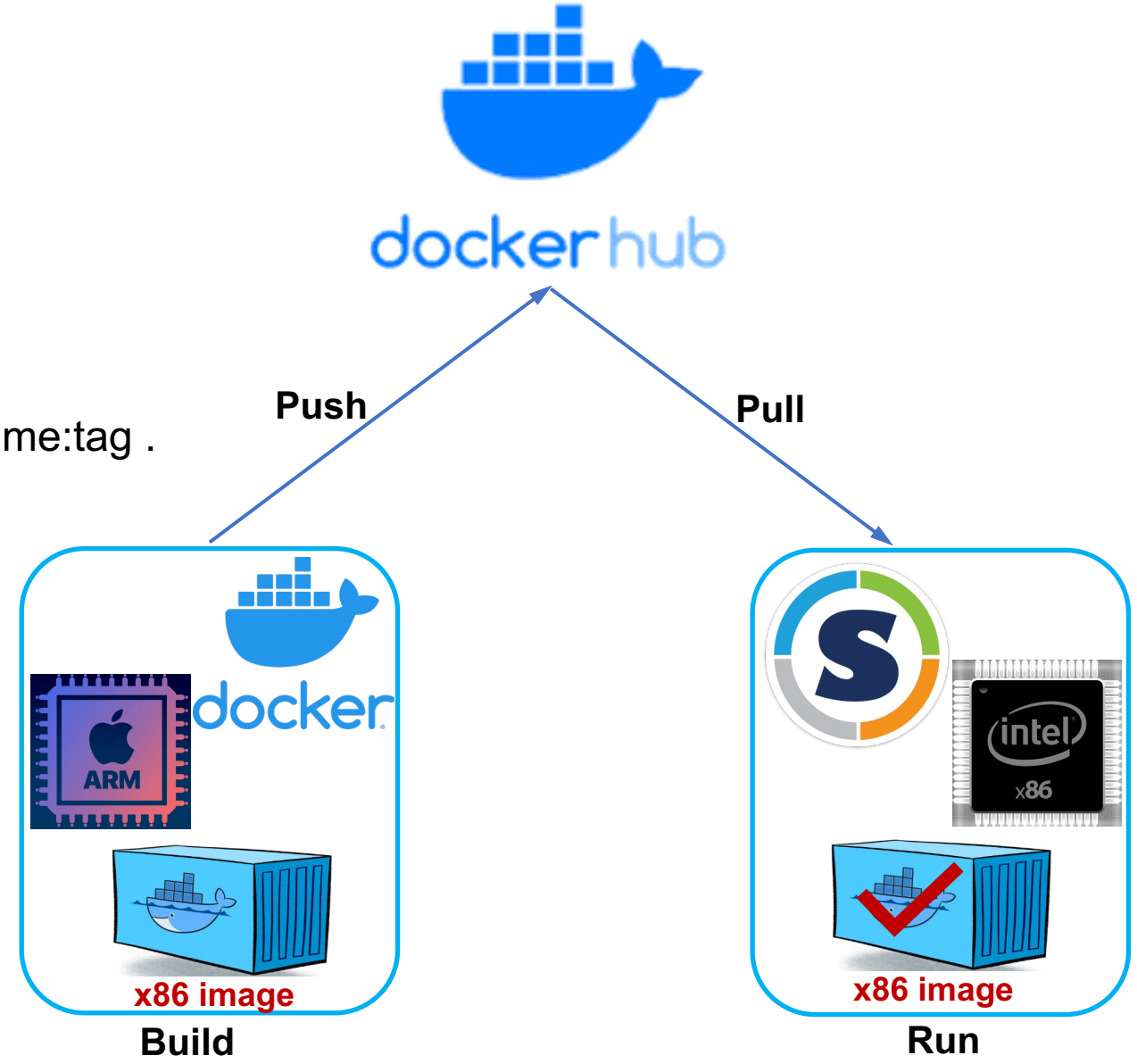




docker build UserName/AppName:tag .



docker build --platform linux/amd64 UserName/AppName:tag .



Thank you

Ticket: tts-research@tufts.edu
Email: yzhang85@tufts.edu
Consultation: https://go.tufts.edu/yucheng

Pulling, Running and Building Containerized HPC Applications

Hands-on Demo

https://zhan4429.github.io/TuftsContainers.github.io/