Using Singularity Containers on the FASRC clusters

Objectives

- Difficulties on HPC systems
- Why use Singularity containers?
- Singularity containers
- How to build your own Singularity containers
- How to run Singularity containers on Cannon/FASSE
- Bind mounts

Difficulties on HPC systems

- Building software is often complicated, particularly on a shared and multi-tenant system
- Some applications might need dependencies that are not readily available and complex to build from source
- Reproducibility:
  - Different researchers may install different versions of an application and/or dependencies
- Portability
  - Hard to share workflows and pipelines, especially with external collaborators who use another HPC system

Why use Singularity containers?

Overcome software stack, reproducibility and portability difficulties

- Create a virtual environment that contains all the software stack needed
- They package in one single file all necessary dependencies
- Choose a (linux) operating system that is different than host (i.e. HPC cluster)
- Easy to publish
- Portable

Virtual machine vs. container

Virtual Machines	Containers
Very flexible -- for example, run Windows on MacOS	Less flexible Only Linux systems
Heavyweight -- need to install all files of virtual environment	Very lightweight -- uses the kernel of host OS

Adapted from LSU Singularity training slides:
http://www.hpc.lsu.edu/training/weekly-materials/2022-Fall/HPC_Singularity_Fall2022.pdf

SingularityCE

- Open-source container software
- Specifically designed for HPC systems (i.e. multi-tenant systems)
  - No root (admin) privileges
- Package applications with their dependencies and workflow into on single file
- Other container software

Charliecloud

SHIFTER

Apptainer

Singularity, SingularityCE, Apptainer

- Singularity: deprecated
- SingularityCE and Apptainer: branches/children of Singularity
- SingularityCE: maintained by Sylabs
- Apptainer: maintained by the Linux Foundation

Docker vs. SingularityCE

- Assumes user has root (admin) privileges on the host system
- Not designed for HPC systems

- Assumes user **does not** have root (admin) privileges on the host system
- Designed for HPC systems

Singularity vocabulary

- SingularityCE or Apptainer – the software
  - As in “SingularityCE 3.11” or “Apptainer 1.0”
- Image
  - a compressed, usually read-only file that contains an OS and specific software stack
  - Examples: “Build a Matlab 2021a image”, “Build an Alphafold image”
- Container
  - The technology: “containers vs. virtual machines”
  - An instance of an image
    - Example: “process my data in a Singularity container of Matlab”
- Host – computer/supercomputer where the image is run

Singularity workflow

1. Build Singularity image (only once) with one of the following methods
    - Pull (i.e. download) existing container from SingularityCE Container Library
    - Pull existing Docker container from DockerHub (downloads as Singularity container)
    - Build a SingularityCE container from a Singularity definition file directly on Cannon/FASSE – unprivileged build with proot
    - Build a SingularityCE container from a local Singularity definition file using option --remote. This will build an image on Sylabs cloud which is automatically downloaded to Cannon/FASSE
2. Use image (many times)

How to build SingularityCE images

- Singularity is only available on compute nodes!!!
  - Cannon: request interactive job using the `salloc` command
  - FASSE: does not allow `salloc` – request a Remote Desktop job on FASSE Open OnDemand and launch a terminal
  - For details, see SingularityCE on the clusters
- Follow docs: https://github.com/fasrc/User_Codes/blob/master/Singularity_Containers/README.md#build-your-own-singularityce-container

Singularity definition file

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

Header: base container image

```
%labels
    Author: J. Harvard
```

Label: container metadata

```
%post
    apt-get -y update
    apt-get -y install cowsay lolcat
```

Post: section where you add your own packages

```
%environment
    export LC_ALL=C
    export PATH=/usr/games:$PATH
```

Environement: set environmental variables

```
%runscript
    date | cowsay | lolcat
```

Runscript: commands run when you use “singularity run”

Unprivileged builds with `proot`

Unprivileged builds that use proot have limitations, because proot's emulation of the root user is not complete. In particular, such builds:

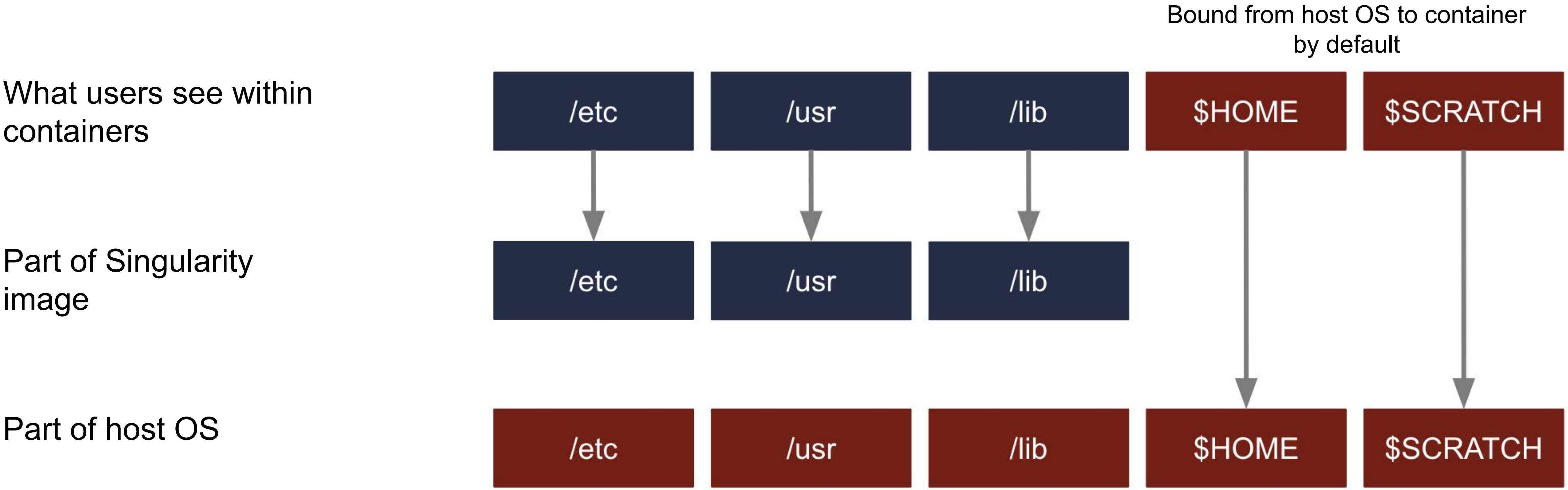
- Header
  - Do not support `arch` / `debootstrap` / `yum` / `zypper` bootstraps
  - Use `localimage`, `library`, `oras`, or one of the `docker`/`oci` sources.
- Do not support `%pre` and `%setup` sections of definition files.
- Run the `%post` sections of a build in the container as an emulated root user.
- Are subject to any restrictions imposed in singularity.conf.
- Incur a performance penalty due to the``ptrace``-based interception of syscalls used by proot.
- May fail if the `%post` script requires privileged operations that proot cannot emulate.

From https://docs.sylabs.io/guides/latest/user-guide/build_a_container.html#unprivilged-proot-builds

How to run Singularity images

- Follow docs: https://github.com/fasrc/User_Codes/blob/master/Singularity_Containers/working_with_images.md

Singularity and host file system



Bound from host OS to container by default

What users see within containers

/etc

/usr

/lib

$HOME

$SCRATCH

Part of Singularity image

/etc

/usr

/lib

Part of host OS

/etc

/usr

/lib

$HOME

$SCRATCH

To allow other filesystems to be accessible from container, you can use `--bind` option

- See Accessing files from a container

Parallel computing and Singularity

- OpenMP
- MPI

Resources and help

- Documentation
  - https://docs.rc.fas.harvard.edu/
  - Singularity docs: https://github.com/fasrc/User_Codes/tree/master/Singularity_Containers
- Portal
  - http://portal.rc.fas.harvard.edu/rcrt/submit_ticket
- Email
  - rchelp@rc.fas.harvard.edu
- Office Hours
  - Wednesday noon-3pm https://harvard.zoom.us/j/255102481
- Consulting Calendar
  - https://www.rc.fas.harvard.edu/consulting-calendar/
- Training
  - https://www.rc.fas.harvard.edu/upcoming-training/

Thank you!