Tutorial: Introduction to Containers for Scientific Container-Native Workflows: 

**Singularity on ACES**

Richard Lawrence  
10/17/2023
Outline

- Overview of Containers
- Overview of Singularity
- Getting Started
- Container Image Sources
- Working with Images
- Working with Containers
- Containerized Scientific Applications on ACES
  - PyTorch
  - LAMMPS
Learning Resources

- Slides on the course web page
  https://hprc.tamu.edu/training/aces_containers_scientific.html
  (highly recommended for working along)
- HPRC’s Knowledge Base
  https://hprc.tamu.edu/kb/Software/Singularity/
- HPRC on YouTube
  https://www.youtube.com/c/TexasAMHPRC
- ACCESS Links
  https://support.access-ci.org/ci-links
Overview of Containers
What Are Containers?

- A container is a process (⚙️) that has its own **view** of local resources:
  - Filesystem
  - User IDs
  - Network
  - etc.
- Example: this container (⚙️ on the right) sees the **image** instead of the physical filesystem
Why Use Containers?

- **Shareability:**
  - Share your container image file by uploading to a public repository
  - Use images shared by others

- **Portability:**
  - Use images on any computer with the same architecture (x86-64)

- **Reproducibility:**
  - Container users are largely unaffected by changes to the cluster environments
What Goes In Container Images?

- Unlike in VMs, the OS Kernel is not duplicated
- Container images are smaller than VM images

Diagram:

- **Local Build, or "Bare metal"**
  - User Application
  - Host Binaries
  - Host Libraries
  - Host OS Kernel
  - Hardware

- **Virtual Machine**
  - User Application
  - Guest Binaries
  - Guest Libraries
  - Guest OS Kernel
  - Virtual Machine Manager

- **Container**
  - User Application
  - Guest Binaries
  - Guest Libraries
  - Container Runtime
  - Host OS Kernel
  - Hardware
Popular Container Runtimes

Instant deployment to users on different devices!

- LXC (2008)
- Docker (2013)
- Singularity (2015)
- Shifter (2016)
- Charliecloud (2017)
- Podman (2018)
Overview of Singularity
Singularity

- An easy-to-use, high-performance container solution

Deploying Secure Container Solutions from Edge to Exascale

Presented by Sylabs
Singularity is Apptainer
Singularity Features

- Singularity is a container runtime and an image builder
- Singularity can read and convert Docker images
- Filesystem inside container is isolated
- User inside container is the same as the user outside
- Works with high-performance cluster technologies

Read more in the Apptainer manual
https://apptainer.org/user-docs/3.8/
Singularity on ACES

- Singularity is available on Compute nodes
  - Singularity activities are too cpu-intensive for login nodes.
- Singularity images can be large on disk. Be aware of your storage quota. (/scratch > /home)
- Some container activities may be too I/O-intense for the shared network filesystem. Be courteous to others and use a local filesystem for large image operations.
Getting Started
ACES Portal

ACES Portal portal-aces.hprc.tamu.edu is the web-based user interface for the ACES cluster.

Open OnDemand (OOD) is an advanced web-based graphical interface framework for HPC users.
Authentication via ClLogon

Log-in using your ACCESS CI credentials.

Select the Identity Provider appropriate for your account.
Get a Shell on ACES

Click on “Clusters” menu → `_aces Shell Access`
Success!

Welcome to the ACES login node.
Set Up Your Tutorial Environment

cd $SCRATCH
mkdir s_tutorial
cd s_tutorial
pwd

export TRAINING=/scratch/training/singularity
ls $TRAINING
Set Up Your Singularity Environment

Get to a compute node from the login node
```
srun --time=120 --mem=4G --pty bash -i
```

Return to your tutorial directory (if necessary)
```
cd $SCRATCH/s_tutorial
```

Set your singularity cache directory for temporary files
```
export SINGULARITY_CACHEDIR=$TMPDIR
```

Connect to the internet for fetching images
```
module load WebProxy
```
Your First Singularity Container

Singularity can fetch an image and launch a shell in one line.

```
singularity shell --help
```

Fetch an image and launch a shell from it

```
singularity shell docker://almalinux:8
cat /etc/redhat-release
exit
```

The ACES compute nodes also have Red Hat Linux installed.

```
cat /etc/redhat-release
```
Congratulations!

Welcome to containers
Container Image Sources
Popular Repositories

The most common repository is:
- Docker Hub

Others repositories include:
- Singularity Hub
- Singularity Library
- NVIDIA GPU Cloud
- Quay.io
- BioContainers

See
https://hprc.tamu.edu/kb/Software/Singularity/Examples/#popular-repositories
Docker Hub Example

Docker Hub repositories are named in the form `<group>/<name>` similar to GitHub.

Each image within a repository has a `<tag>` that describes how and when it was built.

This example is `jupyter/scipy-notebook:latest`
Singularity Pull

Singularity can fetch images from repositories and also convert them to the singularity file format at the same time.

```
singularity pull [target-filename] <source>
```

Where `<source>` refers to something on the internet. The syntax depends on where the source is located.

and `[target-filename]` includes the file extension.
Singularity Pull Example

The `<source>` argument for Docker images looks like

\[
docker://<group>/<name>[:<tag>]
\]

Therefore the pull command for the Jupyter example is,

\[
singularity\ pull\ docker://jupyter/scipy-notebook:latest
\]

(Download now or copy from `$TRAINING`; we will need this later)

The default filename will be `scipy-notebook_latest.sif`
Working with Images
Singularity Image Formats

- Singularity container images come in two main formats:
  1. Directory
  2. Single file. Singularity uses the SIF format for single file images. This is the default.
- The `singularity build` tool can convert images in both formats.
  
  ```bash
  singularity build --help
  ```
- The `--sandbox` option is used to create directory-format images.
Singularity Image Exercise

Singularity pull can fetch an image and write to either file format. *(note the order of the arguments)*

```
singularity pull almalinux.sif docker://almalinux:8
```

Singularity can convert an image to the directory file format. Use the --sandbox argument to specify the directory type. *(note the order of the arguments)*

```
singularity build --sandbox $TMPDIR/almalinux almalinux.sif
```
Singularity Write Exercise

Directory images are writable. Simply add the --writeable flag to your container command.

```
singularity shell --writable $TMPDIR/almalinux
mkdir /my_dir
exit
```

Are the changes still there?
```
singularity shell $TMPDIR/almalinux
ls /
```
Singularity Read-only Exercise

*SIF files are safe for network file system /scratch.*

```
singularity build --fakeroot my_almalinux.sif $TMPDIR/almalinux
```

Are the changes still there?

```
singularity shell my_almalinux.sif
ls /
exit
```

What about the --writeable flag?

```
singularity shell --writable my_almalinux.sif
no.
```
Working with Containers
Launching Processes

Singularity has three methods for launching processes:

- **Interactive**: singularity `shell`
- **Batch processing**: singularity `exec`
- **Container-as-executable**: singularity `run`
Singularity Run Exercise

Singularity run will execute the default runscript, if one was defined. You may also execute the container directly.

```bash
singularity pull docker://hello-world
singularity run hello-world_latest.sif
Hello from Docker!

./hello-world_latest.sif
Hello from Docker!
```

Docker hello-world is a minimal image. This is all it can do.
Singularity Exec Exercise

Singularity Exec lets you access executables and other commands in a container. This is appropriate for batch jobs.

ACES nodes have Python 3.
```
python3 --version
Python 3.8.6
```

Our singularity image has a different Python 3.
```
singularity exec scipy-notebook_latest.sif python3 --version
Python 3.11.6
```
Working with Files

- Filesystem inside a container is isolated from the real, physical filesystem.
- To access your files, ensure the directory is *mounted*.
- By default, Singularity will mount `$HOME` and `$PWD` if it can.
- To specify additional directories, use the `SINGULARITY_BINDPATH` environment variable or the `--bind` command line option.
Working with Files Exercise

Recommended that you mount `/scratch` to get access to your data storage, and `/tmp` to get access to the local disk on the node.

```bash
singularity shell --bind "'/scratch,/tmp""
mkdir $TMPDIR/my_dir; exit
ls $TMPDIR
```

Notice that your variables like `$TMPDIR` get passed into the container by default.

*(singularity on ACES already binds these directories by default)*
Singularity Batch Example

#!/bin/bash

## JOB SPECIFICATIONS
#SBATCH --job-name=sing_test         # Set the job name to "sing_test"
#SBATCH --time=00:10:00              # Set the wall clock limit to 1hr and 30min
#SBATCH --ntasks=4                   # Request 4 task
#SBATCH --mem=2560M                  # Request 2560MB (2.5GB) per node
#SBATCH --output=sing_test.%j        # Send stdout/err to "sing_test.[jobID]"

export SINGULARITY_BINDPATH="/scratch,/tmp"

# execute the default runscript defined in the container
singularity run hello-world_latest.sif

# execute a command within container
# specify the full path if the command is not in PATH
singularity exec scipy-notebook_latest.sif python3 hello.py

ONE VARIABLE
2 CONTAINERS
Interactive Graphical Computing

ACES OnDemand Portal ➔ Interactive Apps ➔ Jupyter Notebook

- Click "Interactive Apps" first.
- Then click "Jupyter Notebook".

Click ➔ click
Containerized Jupyter Notebook

Choose Containers

Enter
$SCRATCH/s_tutorial/scipy-notebook_latest.sif
or wherever your file actually is

Backup copy at
/scratch/training/singularity/scipy-notebook_2023.sif
...Continued

Click
...wait
Click
...wait
Click

WOW
Containerized Scientific Applications
Singularity with GPU

- Containers should be built with CUDA version compatible with local GPUs (CUDA $\geq 11$)
- Just add the `--nv` flag to your singularity command

Many repositories on Docker Hub have GPU-ready images. Search for images with “gpu” in tags

The nvidia cloud also provides GPU-ready images. See: https://hprc.tamu.edu/wiki/SW:Singularity:Examples#NVIDIA_GPU_Cloud
NVIDIA Container Registry Example

```bash
singularity pull docker://nvcr.io/nvidia/pytorch:23.09-py3
```

warning: do not attempt
PyTorch GPU Exercise

Image file: pytorch_23.09-py3.sif from
docker://nvcr.io/nvidia/pytorch:23.09-py3
Located at /scratch/training/singularity/
From the login node: (all on one line)
   srun --mem=4G --time=60
      --gres=gpu:1 --partition=gpu --pty bash -i
From the compute node: (all on one line)
   singularity exec --nv pytorch_23.09-py3.sif
   python3 -c "import torch;
   print(torch.cuda.device_count())"
LAMMPS Molecular Dynamics on GPUs

- LAMMPS is a classical MD code
- [https://www.lammps.org/](https://www.lammps.org/) has a cool animated logo.
LAMMPS on H100 GPUs

- *This* specific build works with H100 GPUs
LAMMPS on GPUs

Located at /scratch/training/singularity/
From the login node: (all on one line)

```bash
srun --mem=4G --time=60
    --gres=gpu:1 --partition=gpu
    --pty bash -i
```

From the compute node:

```bash
cd /scratch/training/singularity
```

(all on one line):

```bash
singularity run --nv lammps-nv-patch_15Jun2023.sif
bash benchmark.sh
```
Acknowledgements

This work was supported by

- the National Science Foundation (NSF), award numbers:
  - 2112356 - ACES - Accelerating Computing for Emerging Sciences
  - 1925764 - SWEETER - SouthWest Expertise in Expanding, Training, Education and Research
  - 2019129 - FASTER - Fostering Accelerated Scientific Transformations, Education, and Research
- Staff and students at Texas A&M High-Performance Research Computing.
- ACCESS CCEP pilot program, Tier-II
Help us help you. Please include details in your request for support, such as, Cluster (Faster, Grace, Terra, ViDaL), NetID (UserID), Job information (Job id(s), Location of your jobfile, input/output files, Application, Module(s) loaded, Error messages, etc), and Steps you have taken, so we can reproduce the problem.