Tutorial: Introduction to Containers for Scientific Container-Native Workflows: **Charliecloud** on **ACES**

Richard Lawrence
9/22/2023

High Performance
Research Computing
DIVISION OF RESEARCH

developed for

Outline

- Overview of Containers
- Overview of Charliecloud
- Getting Started
- Scientific Container Image Sources
- Working with Images
- Working with Containers
- Scientific Use Cases on ACES
  - Tensorflow
  - LAMMPS
  - Clara Parabricks

Learning Resources

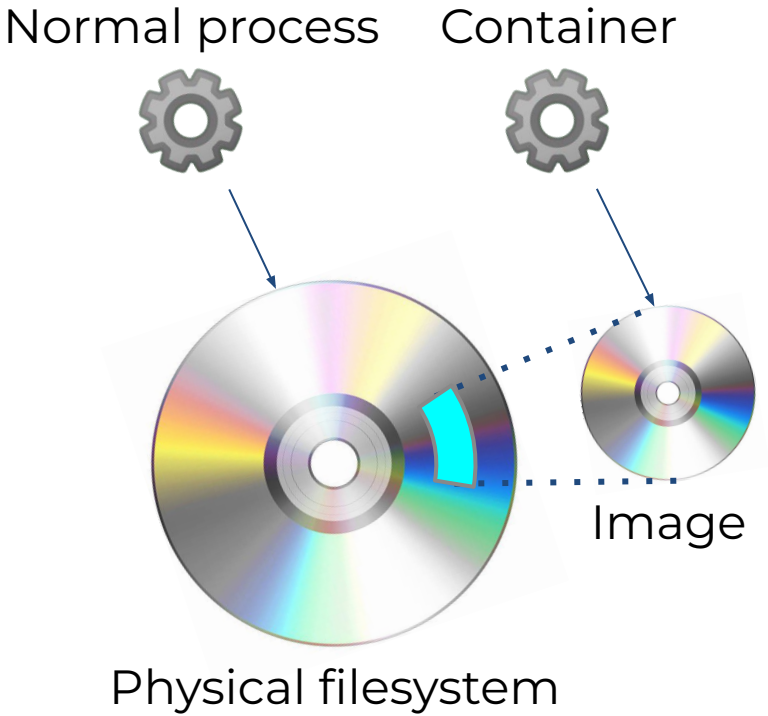
- Slides on the course web page
  https://hprc.tamu.edu/training/aces_containers.html
- HPRC's Knowledge Base
  https://hprc.tamu.edu/kb/Software/CharlieCloud/
- HPRC on YouTube
  https://www.youtube.com/c/TexasAMHPRC
- Charliecloud Documentation
  https://hpc.github.io/charliecloud/
- ACCESS Links
  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

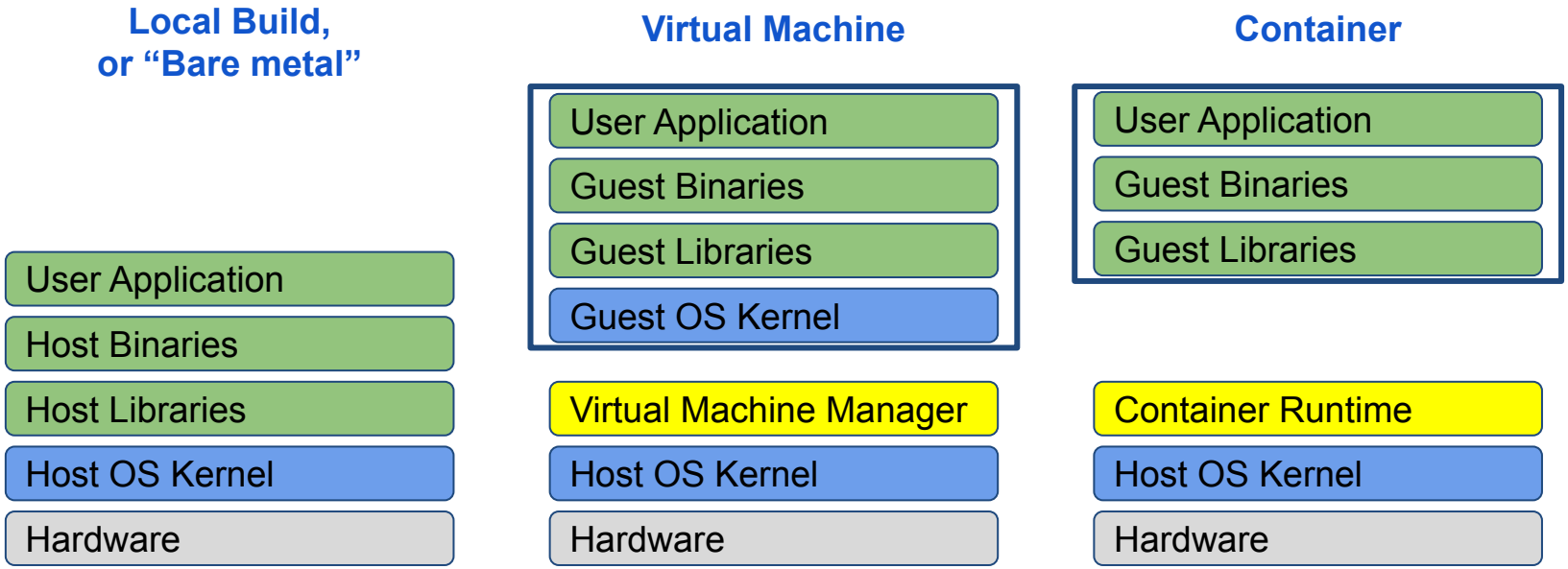


Why Use Containers?

- **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 (x84-64)
- **Reproducibility**:
  - Container users are largely unaffected by changes to the cluster environments

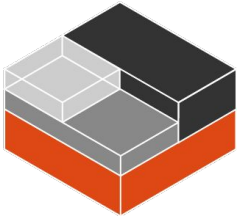
What Goes In Container Images?

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

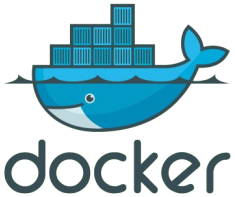


Popular Container Runtimes

Instant deployment to users on different devices!



LXC
2008



Docker
2013

Singularity
2015

Shifter 2016

Charliecloud
2017

Podman
2018

The Privilege Dilemma

- Most software requires root privilege to build and run, including software that is used in container images
- Option 1: user has full root privilege (e.g. Docker)
  - malware can escape and infect the host
  - user can compromise other users' data
  - not safe for shared HPC environment
- Option 2: user has zero root privilege (e.g. Singularity)
  - users not able to build or modify container images
  - some softwares refuse to run at all
  - user cannot adapt images to HPC environment
- Option 3: A more nuanced understanding of privilege

Overview of Charliecloud

Charliecloud

- A lightweight, fully-unprivileged container solution

Presented by

Los Alamos NATIONAL LABORATORY

Charliecloud Philosophy

- Transparent
  - Container images are just regular files
  - Compatible with standard linux tools
- Simple
  - Only implement necessary features
  - Each executable does one thing well
- Zero trust
  - All processes are just user processes
  - Let the Linux Kernel handle security

Charliecloud Features

- Charliecloud is a container runtime and an image builder
- Charliecloud can read and convert Docker images
- Filesystem inside container is isolated
- User inside container is isolated
- Works with high-performance cluster technologies

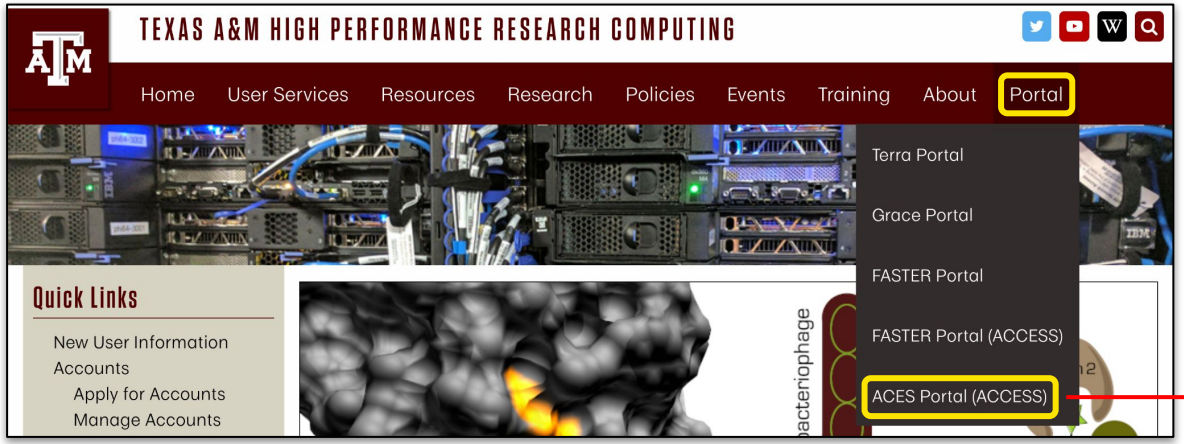
Read more in the Charliecloud manual on github
https://hpc.github.io/charliecloud/

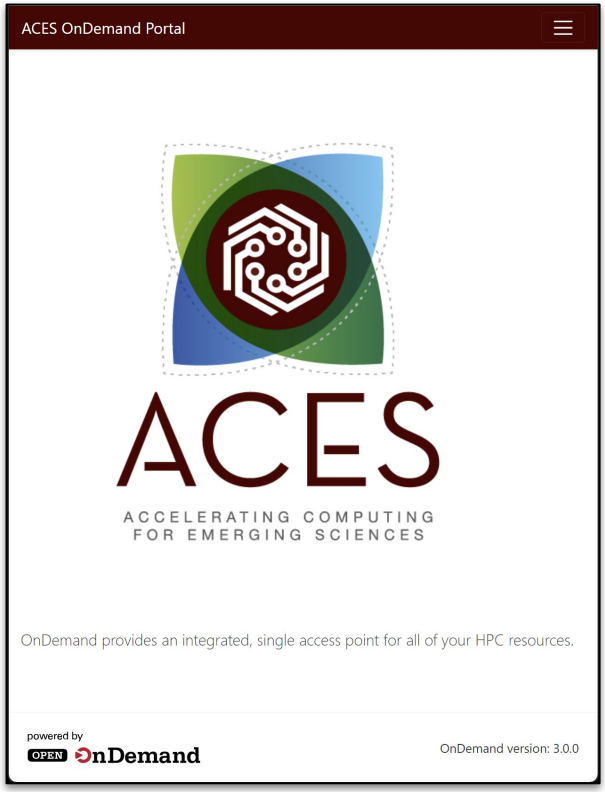
Charliecloud on ACES

- Charliecloud is available from our module system
  - execute `module load charliecloud`
- Charliecloud images can be large on disk. Be aware of your storage quota.
- Some container activities may be too cpu-intense for the *shared* login node. Be courteous to others and use a compute node for large image operations.
- 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



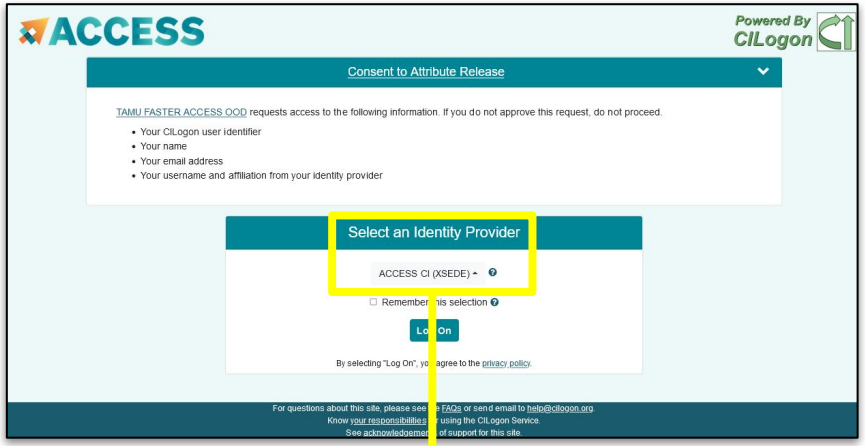


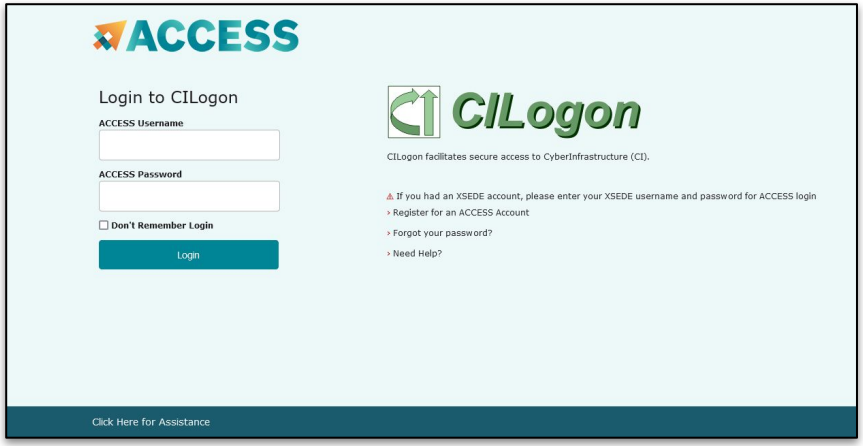
ACES Portal portal-aces.hprc.tamu.edu
is the web-based user interface for the ACES cluster

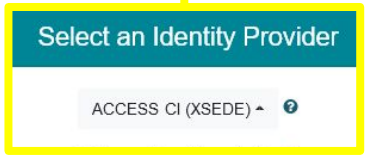
Open OnDemand (OOD) is an advanced web-based graphical interface framework for HPC users

Authentication via CILogin

Log-in using your ACCESS CI credentials.



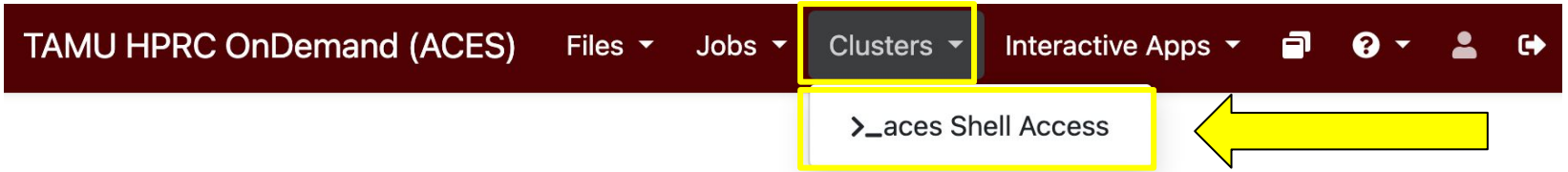




Select the Identity Provider appropriate for your account.

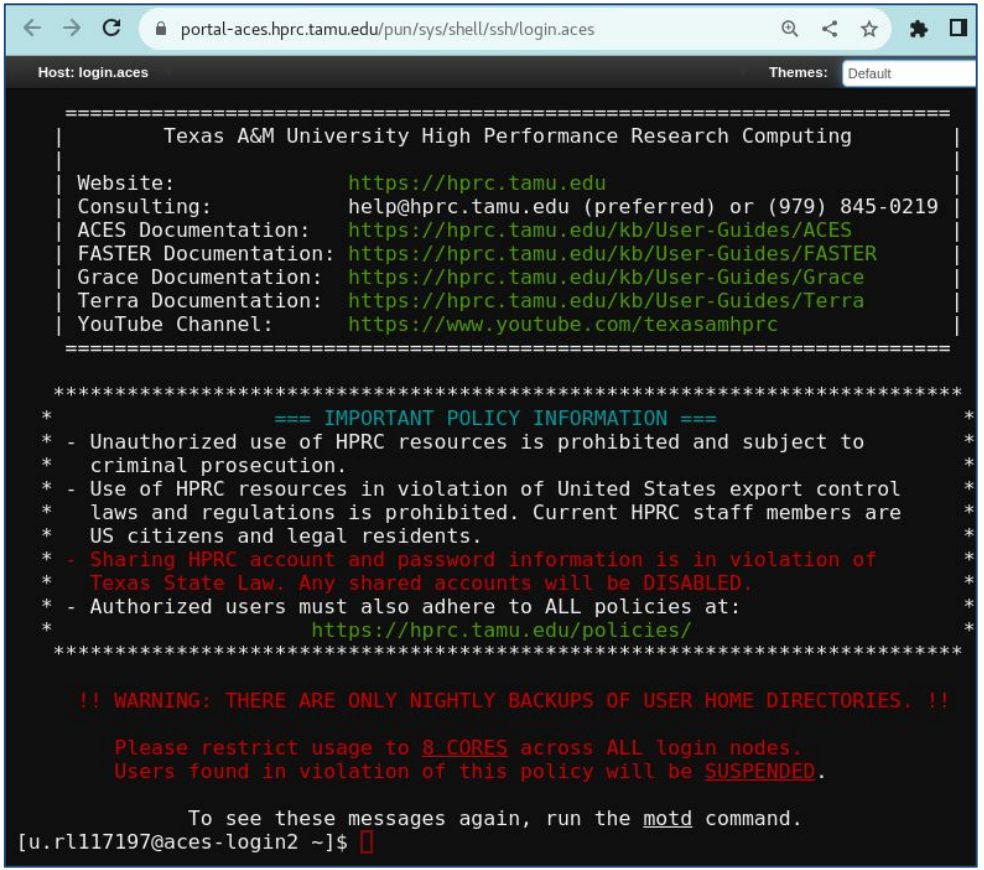
Get a Shell on ACES

Click on “Clusters” menu → _aces Shell Access



Success!

Welcome to the ACES login node.



Set Up Your Environment

```
cd $SCRATCH
mkdir ch_tutorial
cd ch_tutorial
pwd

export TRAINING=/scratch/training/charliecloud
ls $TRAINING

module load charliecloud
module list
```

Your First Image

The charliecloud image tool helps you build and organize your images.

```
ch-image --help
```

Let's fetch a small, basic linux distro: Almalinux.

```
ch-image pull almalinux:8
ch-image list
```

The image is in your personal temporary local image repository.

```
echo $CH_IMAGE_STORAGE
ls $CH_IMAGE_STORAGE/img/
```

Your First Container

The ACES login node has Red Hat Enterprise linux installed.

```
cat /etc/redhat-release
```

The charliecloud-run tool launches containers out of existing images.

```
ch-run --help
```

Let's launch a bash shell, investigate, and stop the container.

```
ch-run almalinux:8 bash
cat /etc/redhat-release
exit
```


Congratulations!
Welcome to containers
WWW.FUNIMADA.COM

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

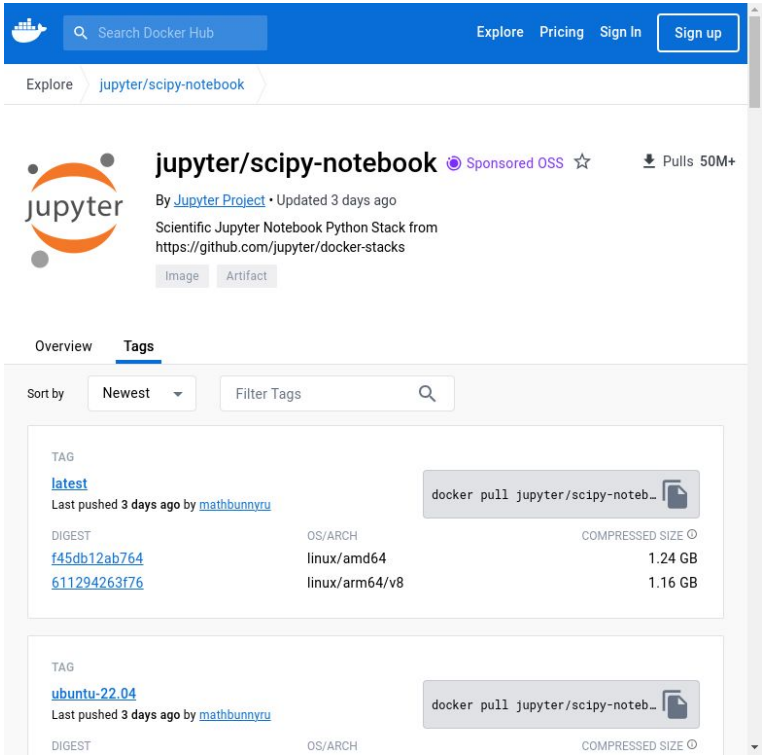
See
https://hprc.tamu.edu/kb/Software/Singularity/Examples/#popular-repositories

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`



Docker Hub Pull Exercise

The `<source>` argument for an image pull looks like

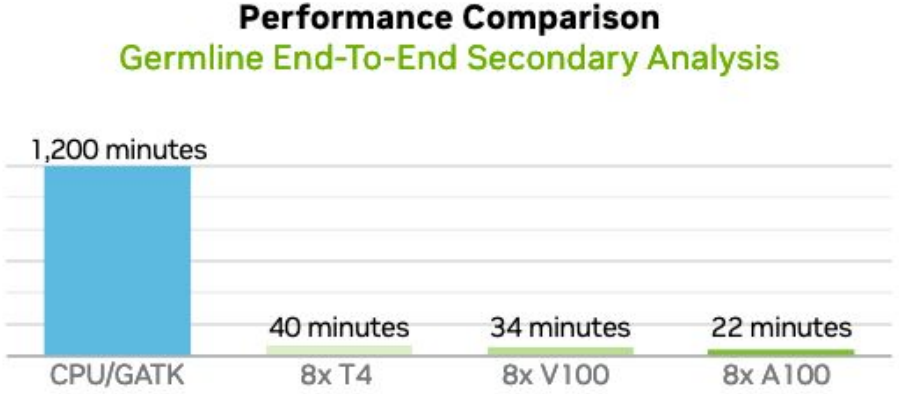
- `<url>/<group>/<name>[:<tag>]`
- No url is needed for this exercise because Docker Hub is the default repository for Charliecloud

Therefore the pull command based the previous slide is:

```
ch-image pull jupyter/scipy-notebook:latest
```

(we will use this one later)

Clara Parabricks for GPU from NVIDIA

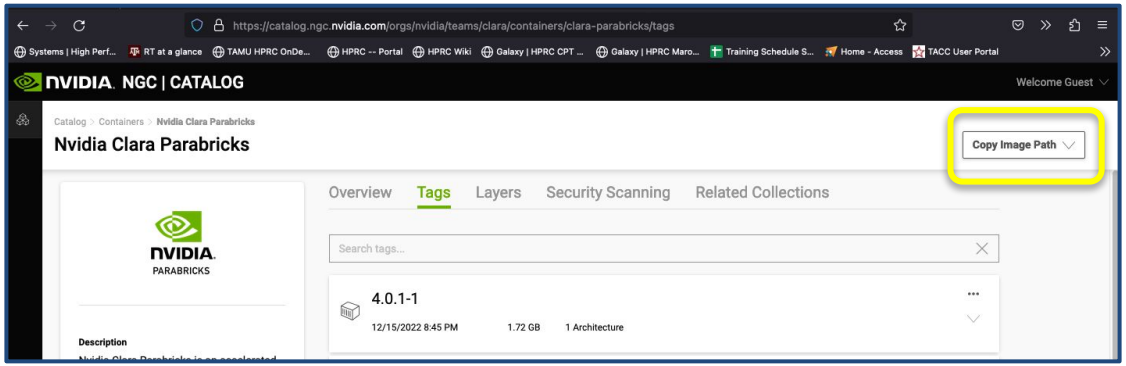


Data was generated using publicly available data (https://precision.fda.gov/challenges/truth) for NA12878, deprecating the data to 30X coverage. For the 22-minute runtime, DGX A100 with 320G memory was used. The native GATK4.1 numbers were generated using 32 vCPU (3.1 GHz Intel Xeon® Platinum 8175M) using 320Gb RAM.

NVIDIA Product Sheet:
https://resources.nvidia.com/en-us-genomics-ug-ep/healthcare-genomics-?lx=M-s96I&ncid=em-nurt-521116&mkt_tok=MTU2LU9GTi03NDIAAAGG5gQCuzMHKWvhCg5ODJ9NTi9KCxm57Lxjd5DcahRJvhUUc-g_yTLDcNVB3HBmOyWbGWigpg4yq1h3SK9QNOLnbLU6cm8VhMCHmup4BGcunnUvwRCy#cid=ix09_em-nurt_en-us

NVIDIA Repository Example

Navigate to catalog.ngc.nvidia.com and search “Clara Parabricks”.



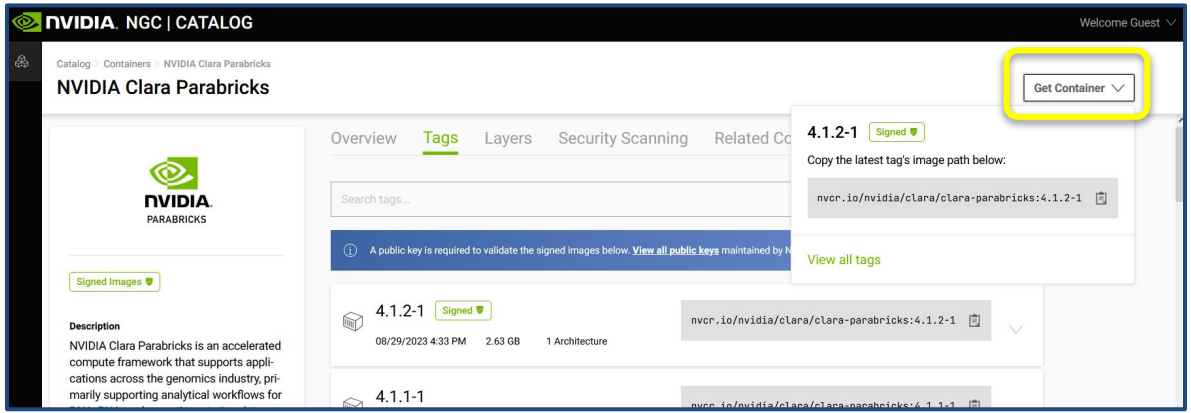
Private repositories need a url in addition to the image name.

```
<url>/<group>/<name>[:<tag>]
```

Click the “Copy Image Path” button on the Tags tab.

NVIDIA Repository Example

Navigate to catalog.ngc.nvidia.com and search “Clara Parabricks”.



Private repositories need a url in addition to the image name.

```
<url>/<group>/<name>[:<tag>]
```

Click the “Get Container” button on the Tags tab and copy the path.

Working with Images

Image Formats

- Charliecloud container images come in two main formats:
  1. Directory
  2. Single file. HPRC supports the squashfs filesystem format for single file images. (more about that on a later slide)
- The `ch-convert` tool copies images into different formats

```
ch-convert --help
```

Directory Image Format

- The image name should end in `/`.
- Directory images are writable.
- Directory read/write operation are slow, so put directory images on the high-speed `/tmp` filesystem.
- Images in `$CH_IMAGE_STORAGE` are also directory images, but you refer to them by name without the trailing slash.

Convert to Directory Exercise

Convert our image in the cache to a directory image.

```
mkdir /tmp/$USER
ch-convert jupyter/scipy-notebook:latest /tmp/$USER/jupyter/
```

What did we make?

```
ls /tmp/$USER/jupyter/
```

Editing Images Exercise

Directory images can be modified by adding the

```
--write
```

flag to `ch-run`. Any changes you make will be saved.

```
ch-run --write /tmp/$USER/jupyter/ bash
mkdir /scratch
exit
```

Are the changes still there?

```
ch-run /tmp/$USER/jupyter/ bash
ls
```

Squashfs Image Format

- Squashfs is an open-source file format for filesystem images
- The whole filesystem becomes one single file
- The image name should end in `.sqfs`
- Squashfs images are read-only.
- Squashfs read operations are fast, so put squashfs images on the network filesystem `/scratch`.

Convert to Squashfs Exercise

Make sure you are still in your `ch_tutorial` directory in `$SCRATCH`

```
pwd
```

Then convert

```
ch-convert /tmp/$USER/jupyter/ jupyter.sqfs
```

Are your changes still there?

```
ch-run jupyter.sqfs /bin/bash
ls
exit
```

Working with Containers

Mounting your Scratch Space

- The option `-b` is used to mount the `/scratch` filesystem outside the container over the empty `/scratch` directory inside the container.
- The option `-c` is used to set the starting working directory in the container.

```
ch-run -b /scratch -c $SCRATCH jupyter.sqfs bash
pwd
ls
exit
```

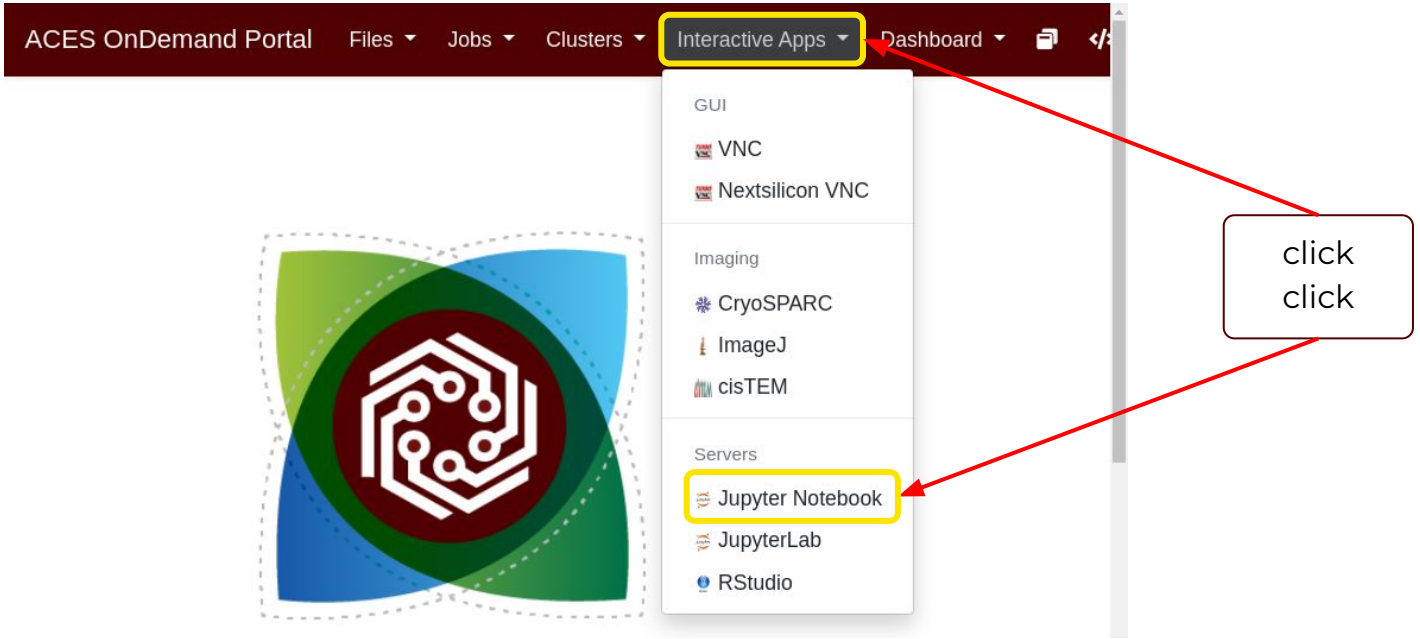
Working with Variables

Some containers come with environment variables that are needed in order for the application to function properly. The `--set-env` option is used to turn those on.

```
ch-run --set-env jupyter.sqfs python
>>> import numpy
>>> print(numpy)
>>> exit()
```

Python with Numpy was installed in a Conda environment. It requires the `PYTHONPATH` variable to function.

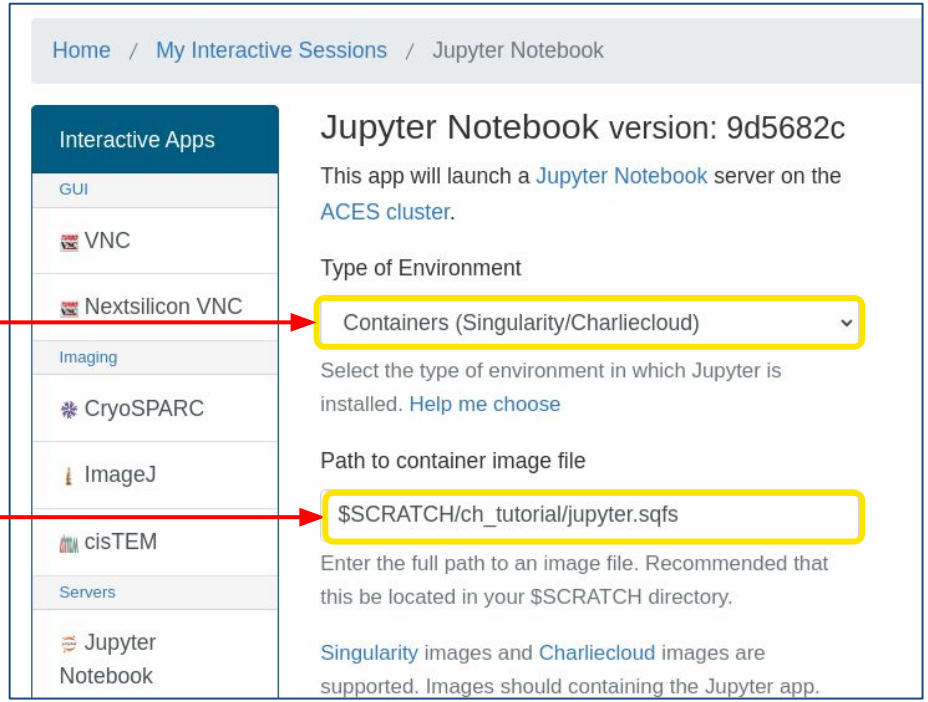
Interactive Graphical Computing



Containerized Jupyter Notebook

Choose *Containers*

Enter `$SCRATCH/ch_tutorial/jupyter.sqfs` or wherever your file actually is



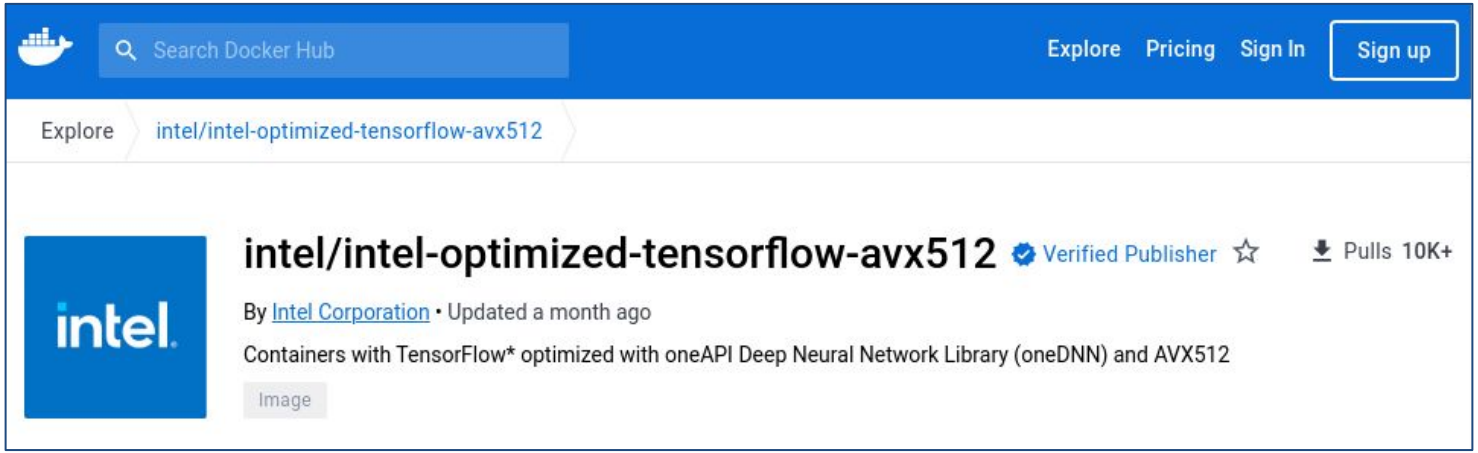
Backup copy at
`/scratch/training/charliecloud/jupyter-scipy-notebook-2023.sqfs`

...Continued


Launch
Jupyter Notebook (5488) 1 node | 1 core | Starting
Jupyter Notebook (5489) 1 node | 1 core | Running
Host: ac110
Delete
Created at: 2023-09-21 15:39:52 CDT
Time Remaining: 56 minutes
Session ID: a5f41dfd-7c0d-44e3-aea7-7331c66a4d24
Connect to Jupyter
click
...wait
click
...wait
click
New
Upload
Notebook
Terminal
Console
New File
New Folder
last month
File Edit View Run Kernel Settings Help
Trusted
Code
[1]:
import numpy
print(numpy)
WOW
<module 'numpy' from '/opt/conda/lib/python3.11/site-packages/numpy/__init__.py'>

Containerized Scientific Applications

Machine Learning with TensorFlow



Pull this intel-optimized image and convert it to Squashfs

```
ch-image pull intel/intel-optimized-tensorflow-avx512
ch-convert intel/intel-optimized-tensorflow-avx512 intel-tensorflow.sqfs
```

TensorFlow in Container

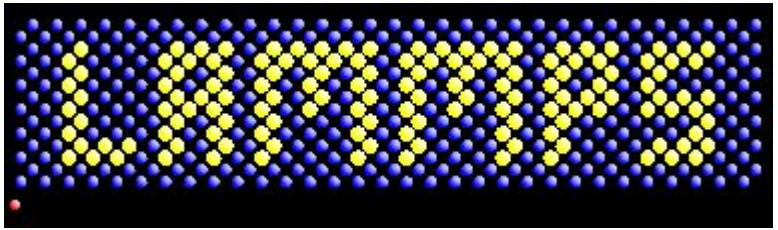
Run the container and import TensorFlow:

```
ch-run intel-tensorflow.sqfs python
Python 3.10.6 …
>>> import tensorflow as tf
…
>>> print("TensorFlow version:", tf.__version__)
TensorFlow version: 2.13.0
>>> exit()
```

Backup copy at
`/scratch/training/charliecloud/intel-tensorflow.sqfs`

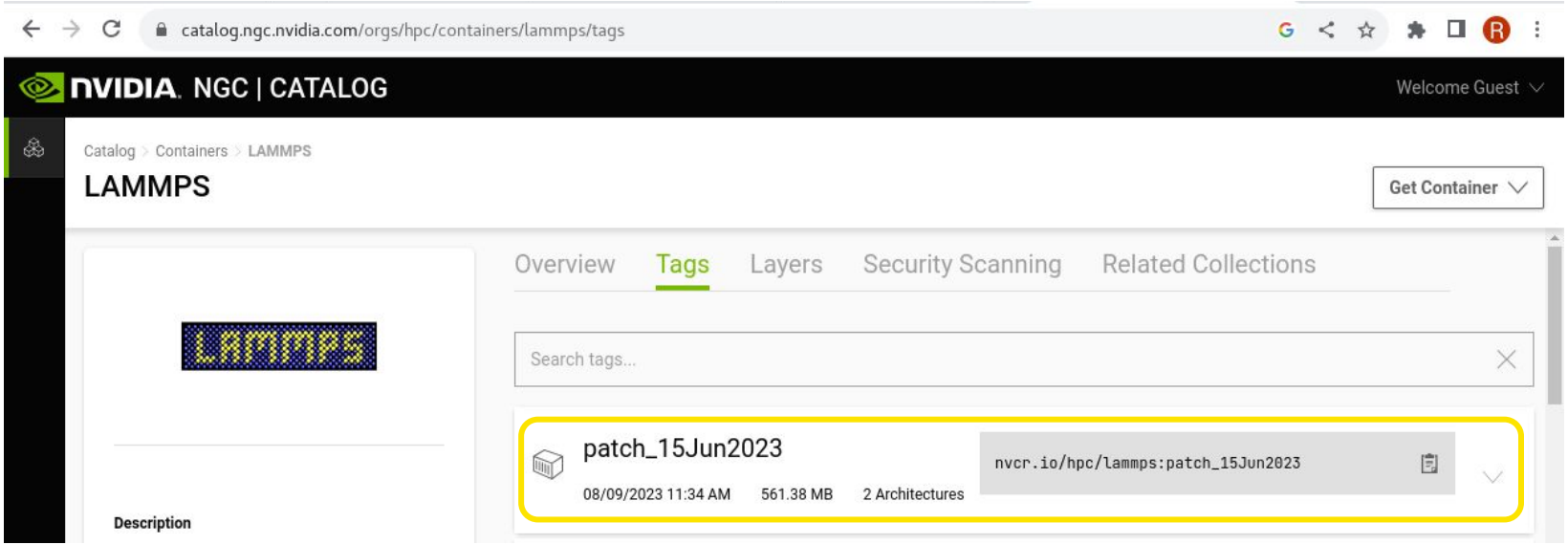
LAMMPS Molecular Dynamics on GPUs

- LAMMPS is a classical MD code
- https://www.lammps.org/ has a cool animated logo.
- NVIDIA provides GPU-ready container images for lammps. https://catalog.ngc.nvidia.com/orgs/hpc/containers/lammps



LAMMPS on H100 GPUs

- *This* specific build works with H100 GPUs



Using GPUs with Charliecloud

- Need to “inject” two things into the container
  1. nvidia libraries and executables
  2. the nvidia runscript
- Tools needed to do the injection
  a. NVIDIA `nvidia-container-cli` tool
  b. charliecloud `ch-fromhost` tool

- On ACES:
  - nvidia-container-cli is provided as a module
  - *Compute nodes* with GPUs have the nvidia libraries
  - We have a copy of the runscript in `$TRAINING`

Get to a Compute Node

(All on one line):

```
srun --mem=240G --time=01:00:00 --gres=gpu:h100:1
    --partition=gpu --cpus-per-task=32 --pty bash
```

Check if you are now on a compute node?

```
hostname
```

Does it have GPUs?

```
nvidia-smi
```

Set up your environment (again)

```
cd $SCRATCH/ch_tutorial
module load charliecloud nvidia-container-cli WebProxy
```

Build a GPU-ready image

(on compute node):

```
ch-image pull nvcr.io/hpc/lammps:patch_15Jun2023
ch-convert nvcr.io/hpc/lammps:patch_15Jun2023 $TMPDIR/lammps
ch-fromhost --nvidia $TMPDIR/lammps
ch-fromhost -d / -p $TRAINING/runscript $TMPDIR/lammps
ch-convert $TMPDIR/lammps lammps.sqfs
```

Note: `$TMPDIR` is a location in `/tmp` that's specific to compute nodes.

LAMMPS in Container

We can now test the container:

```
ch-run --set-env lammps.sqfs -- /runscript mpirun lmp -h
```

Notes: `mpirun` is used to execute LAMMPS to work around a problem with `srun`. `lmp` is the LAMMPS executable

Backup copy at
`/scratch/training/charliecloud/lammps_nv_patch_15Jun2023.sqfs`

LAMMPS on GPUs

Now that we know the container works, we can run a benchmarking example provided by LAMMPS:

```
cp $TRAINING/in.lj.txt .
cp $TRAINING/benchmark.sh .
```

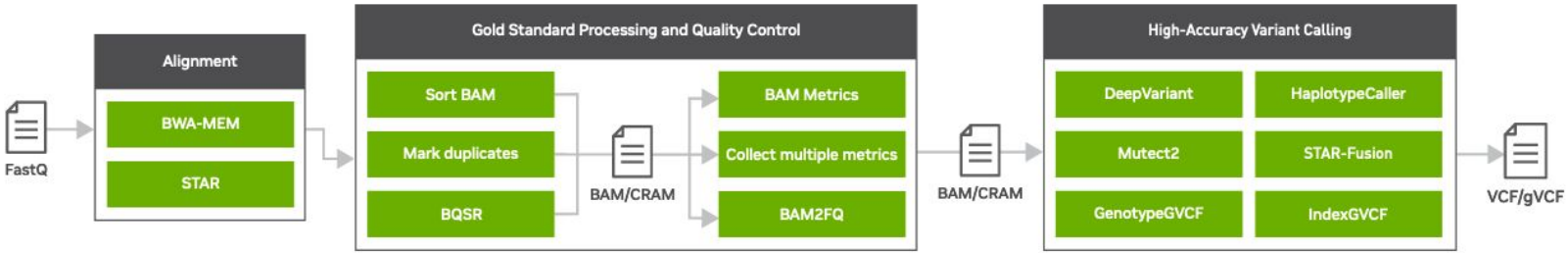
(all on one line)

```
ch-run --set-env -b "$PWD:/host_pwd" -c /host_pwd
lammps.sqfs -- /runscript bash benchmark.sh
```

Backup copy at
`/scratch/training/charliecloud/lammps_nv_patch_15Jun2023.sqfs`

Genomic Analyses with NVIDIA's Clara Parabricks

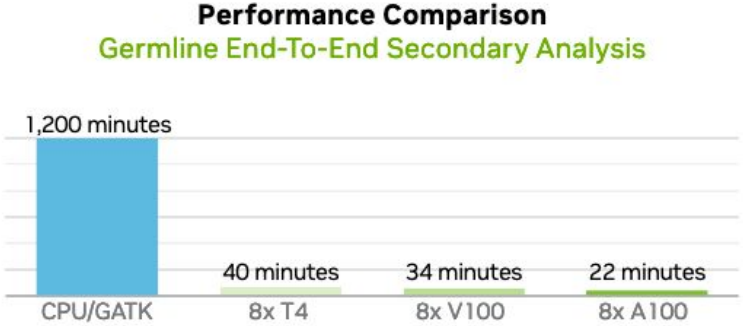
- GPU-accelerated version of common bioinformatics pipeline
- Works with both RNA-seq and WGS data
- NVIDIA provides images that containers easily integrate with Charliecloud
- Today's exercise will focus on completing the first portion of the pipeline



Parabricks

NVIDIA

Genomic Analyses with NVIDIA's Clara Parabricks



Data was generated using publicly available data (https://precision.fda.gov/challenges/truth) for NA12878, deprecating the data to 30X coverage. For the 22-minute runtime, DGX A100 with 320G memory was used. The native GATK4.1 numbers were generated using 32 vCPU (3.1 GHz Intel Xeon® Platinum 8175M) using 320Gb RAM.

Parabricks

NVIDIA

Get to a Compute Node (reminder)

Reminder: if you aren't on a compute node,
(all on one line)

```
srun --mem=240G --time=01:00:00 --gres=gpu:h100:1
        --partition=gpu --cpus-per-task=48 --pty bash
```

Followed by:

```
module load charliecloud nvidia-container-cli WebProxy
```

Genomic Analyses Example Files

Make a subdirectory

```
cd $SCRATCH/ch_tutorial
mkdir ch_parabricks
cd ch_parabricks
```

Copy the example material

```
cp $TRAINING/sample* .
cp $TRAINING/Homo* .
ls
```

Build a GPU-ready Clara Parabricks Image

Pull the parabricks image from NVIDIA using Charliecloud:
(all on one line)

```
ch-image pull
nvcr.io/nvidia/clara/clara-parabricks:4.1.1-1
parabricks-4.1.1-1
```

Build the GPU-ready image

```
ch-convert parabricks-4.1.1-1 $TMPDIR/parabricks4.1
ch-fromhost --nvidia $TMPDIR/parabricks4.1
ch-convert $TMPDIR/parabricks4.1 parabricks4.1.sqfs
```

NVIDIA's Clara Parabricks in Container

- Now we are ready to run Parabricks!

(all on one line)

```
ch-run -b "$PWD:/mnt/1" -c "mnt/1" parabricks4.1.sqfs
        pbrun fq2bam -- --ref Homo_sapiens_assembly38.fasta
        --in-fq sample_1.fastq.gz sample_2.fastq.gz --out-bam test.bam
```

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

High Performance Research Computing

DIVISION OF RESEARCH

https://hprc.tamu.edu

HPRC Helpdesk:

help@hprc.tamu.edu

Phone: 979-845-0219

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.