University of Colorado **Anschutz Medical Campus**

Container workshop

By Kevin Fotso

Where are the container software?

There are two ways:

- Apptainer, which is installed at a system level on any compute partition (e.g. acompile, atesting, amilan etc…) .

Where are the container software?

There are two ways:

- Apptainer, which is installed at a system level on any compute partition (e.g. acompile, atesting, amilan etc…) .

```
~]$  which apptainer
/usr/bin/apptainer
```

Where are the container software?

There are two ways:

- Apptainer, which is installed at a system level on any compute partition (e.g. acompile, atesting, amilan etc…) .

```
~]$  which apptainer
/usr/bin/apptainer
```

- Singularity, which needs to be loaded using LMOD at a system level on any compute partition (e.g. acompile, atesting, amilan etc…) .

Where are the container software?

There are two ways:

- Apptainer, which is installed at a system level on any compute partition (e.g. acompile, atesting, amilan etc…) .

```
~]$  which apptainer
/usr/bin/apptainer
```

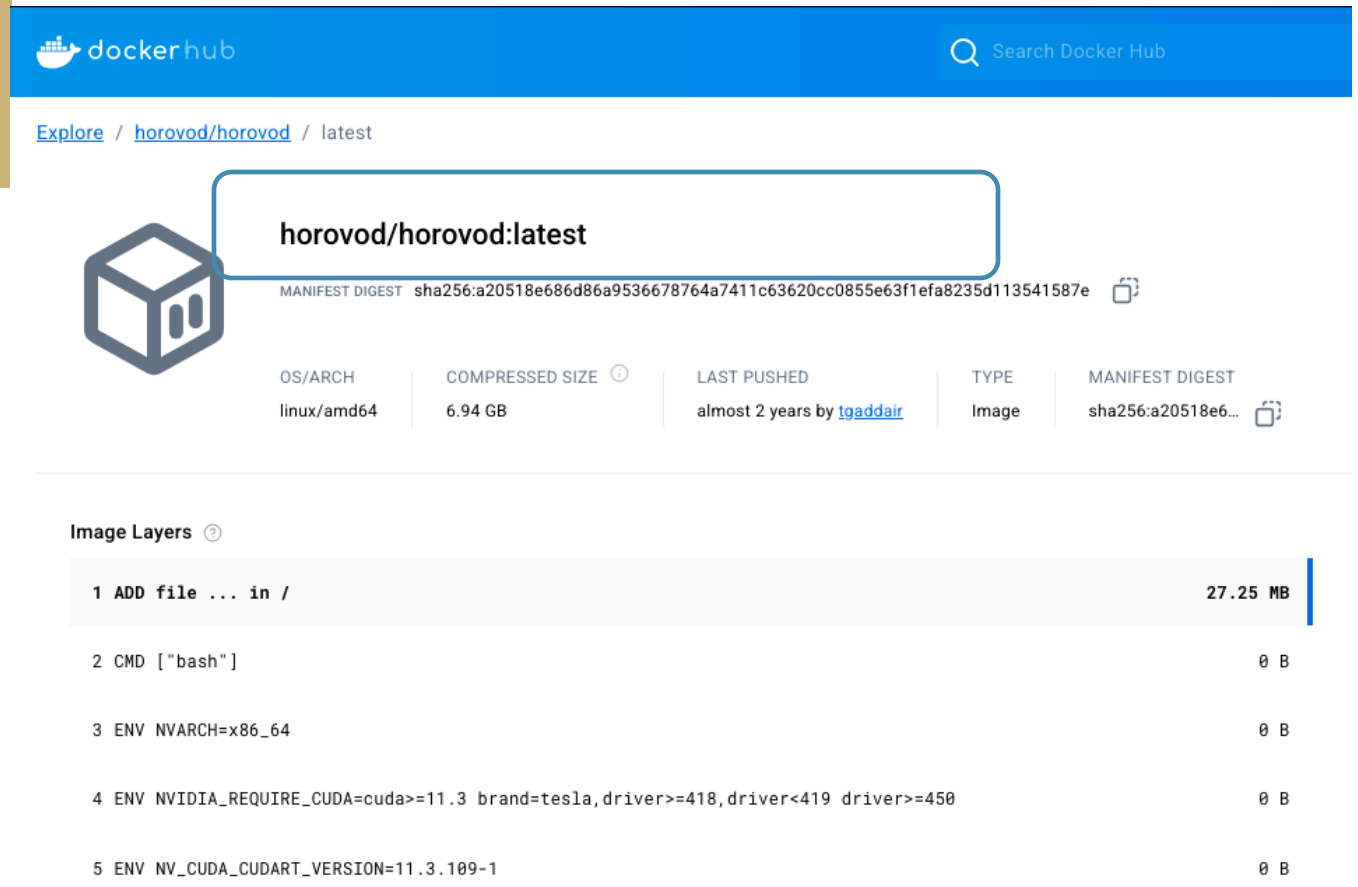
- singularity, which needs to be loaded using at a system level on any compute partition (e.g. acompile, atesting, amilan etc…) .

```
~]$  module avail singularity

singularity/3.6.4 (D)        singularity/3.7.4

~]$  module load singularity

~]$  module list

Currently loaded modules:
1)  slurm/alpine 2) curc-quota/latest 3) StdEnv
4) singularity/3.6.4
```

What are apptainer vs singularity?

- They originally came from the same project called singularity under the Sylabs umbrella.
- A fork occurred in 2020 which eventually led to SingularityCE for Sylabs and apptainer under the Linux Foundation umbrella
  - Source: https://groups.google.com/a/lbl.gov/g/singularity/c/UbywVHXD_co/m/bGa8M4QAAQAJ
  - https://groups.google.com/a/lbl.gov/g/singularity/c/lRWR3vJGvks/m/bMvv9OfYBAAJ?pli=1

Let’s use a Docker container!



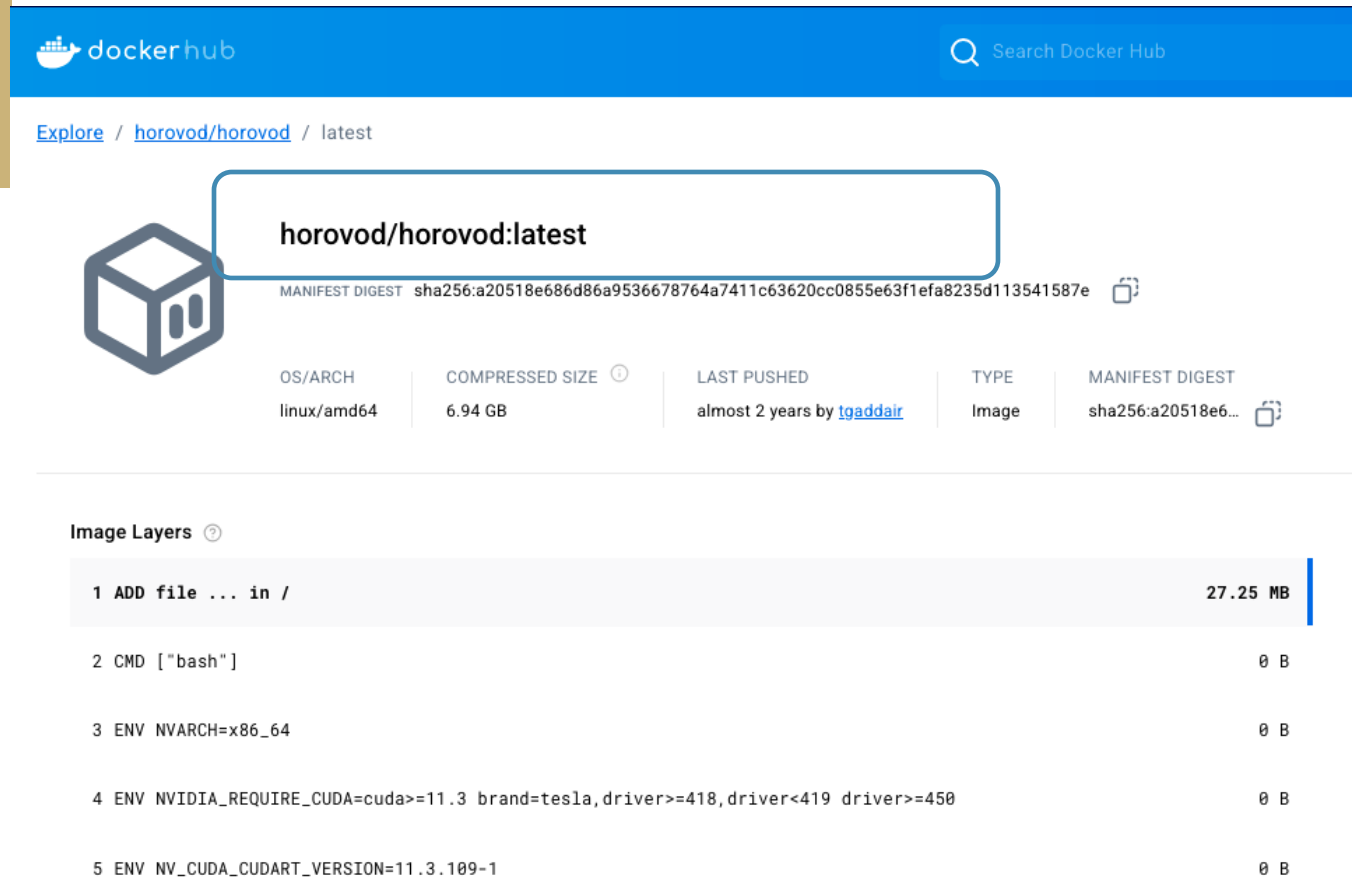
- Horovod allows to run distributed deep learning on multiple cpus, gpus and nodes.

University of Colorado
Anschutz Medical Campus

Source:

https://hub.docker.com/layers/horovod/horovod/latest/images/sha256-a20518e686d86a9536678764a7411c63620cc0855e63f1efa8235d113541587e?context=explore

Let’s use a Docker container!



- Horovod allows run distributed deep learning on multiple cpus, gpus and nodes.

- Please refer to our workshop here:

https://github.com/kf-cuanschutz/CU-Anschutz-HPC-documentation/blob/main/Workshops/Introduction_to_Horovod_102423_part1_official_v2.pdf

University of Colorado
Anschutz Medical Campus

Source:

https://hub.docker.com/layers/horovod/horovod/latest/images/sha256-a20518e686d86a9536678764a7411c63620cc0855e63f1efa8235d113541587e?context=explore

Let’s use a Docker container!

- We make sure to go to a compute node on Alpine

```
@login-ci1 ~]$ acompile –ntasks=4 –time=12:00:00

@c3cpu-a2-u32-2 ~]$
```

Let’s use a Docker container!

- We make sure to go to a compute node on Alpine

```
@login-ci1 ~]$ acompile –ntasks=4 –time=12:00:00

@c3cpu-a2-u32-2 ~]$
```

- Because the filesystem /tmp is small on Alpine, we probably want to export tmp directories to the scratch filesystem.

```
@c3cpu-a2-u32-2 ~]$ export ALPINE_SCRATCH=/gpfs/alpine1/scratch/$USER
@c3cpu-a2-u32-2 ~]$ export APPTAINER_TMPDIR=$ALPINE_SCRATCH/apptainer/tmp
@c3cpu-a2-u32-2 ~]$ export APPTAINER_TMPDIR=$ALPINE_SCRATCH/apptainer/cache
@c3cpu-a2-u32-2 ~]$ mkdir -pv $APPTAINER_TMPDIR $APPTAINER_TMPDIR
```

We convert the docker img into apptainer

- We use the build command, which works equally for apptainer or singularity.

```
@c3cpu-a2-u32-2 ~]$ apptainer build horovod.sif docker://horovod/horovod:latest
INFO: Starting build...
Copying blob 326d76058f67 skipped: already exists
Copying blob 846c0b181fff skipped: already exists
Copying blob 6fc9dd88827c skipped: already exists
```

We can also build a container image out of a def file

- We create what is called a definition file, which will create the necessary instructions to build the container image

This section helps to create the base OS I want.

```
Bootstrap: debootstrap
OSVersion: focal
MirrorURL: http://us.archive.ubuntu.com/ubuntu/
Include: apt
%labels
   Author Kevin Fotso
   Version v1.0.0
%files
   /home/fotsok/spliceAI/SpliceAI /NGS_tools/
%post
   # non-interactive debconf
   export DEBIAN_FRONTEND=noninteractive
DEBCONF_NONINTERACTIVE_SEEN=true
   # update apt
   apt-get update
   # install python3 and snakemake
   apt update
   apt install -y     wget \
            gfortran \
```

We can also build a container image out of a def file

- We create what is called a definition file, which will create the necessary instructions to build the container image

To give some comments and select the package manager toolbox associated with the OS

```
Bootstrap: debootstrap
OSVersion: focal
MirrorURL: http://us.archive.ubuntu.com/ubuntu/
Include: apt
%labels
   Author Kevin Fotso
   Version v1.0.0
%files
   /home/fotsok/spliceAI/SpliceAI /NGS_tools/
%post
   # non-interactive debconf
   export DEBIAN_FRONTEND=noninteractive
DEBCONF_NONINTERACTIVE_SEEN=true
   # update apt
   apt-get update
   # install python3 and snakemake
   apt update
   apt install -y     wget \
            gfortran \
```

We can also build a container image out of a def file

- We create what is called a definition file, which will create the necessary instructions to build the container image

%files allow to copy files from the host system into the container during the build process.

```
Bootstrap: debootstrap
OSVersion: focal
MirrorURL: http://us.archive.ubuntu.com/ubuntu/
Include: apt
%labels
   Author Kevin Fotso
   Version v1.0.0
%files
   /home/fotsok/spliceAI/SpliceAI /NGS_tools/
%post
   # non-interactive debconf
   export DEBIAN_FRONTEND=noninteractive
DEBCONF_NONINTERACTIVE_SEEN=true
   # update apt
   apt-get update
   # install python3 and snakemake
   apt update
   apt install -y     wget \
            gfortran \
```

We can also build a container image out of a def file

- We create what is called a definition file, which will create the necessary instructions to build the container image

```
Bootstrap: debootstrap
OSVersion: focal
MirrorURL: http://us.archive.ubuntu.com/ubuntu/
Include: apt
%labels
   Author Kevin Fotso
   Version v1.0.0
%files
   /home/fotsok/spliceAI/SpliceAI /NGS_tools/
%post
   # non-interactive debconf
   export DEBIAN_FRONTEND=noninteractive
DEBCONF_NONINTERACTIVE_SEEN=true
   # update apt
   apt-get update
   # install python3 and snakemake
   apt update
   apt install -y     wget \
            gfortran \
```

This section allows to pass write config files, create new directories, variables.

Noninteractive is very suitable for automatic build and and to have zero interaction while installing.

We can also build a container image out of a def file

- We create what is called a definition file, which will create the necessary instructions to build the container image

```
Bootstrap: debootstrap
OSVersion: focal
MirrorURL: http://us.archive.ubuntu.com/ubuntu/
Include: apt
%labels
   Author Kevin Fotso
   Version v1.0.0
%files
   /home/fotsok/spliceAI/SpliceAI /NGS_tools/
%post
   # non-interactive debconf
   export DEBIAN_FRONTEND=noninteractive
DEBCONF_NONINTERACTIVE_SEEN=true
   # update apt
   apt-get update
   # install python3 and snakemake
   apt update
   apt install -y     wget \
            gfortran \
```

Here, I can install the packages

We can also build a container image out of a def file

- We create what is called a definition file, which will create the necessary instructions to build the container image

Here, I can export other environmental variables

```
%environment
export CONDA_PREFIX='/opt/software/conda'
export PATH='/opt/software/conda/bin':$PATH
export LD_LIBRARY_PATH=$CONDA_PREFIX/lib/python3.10/site-packages/nvidia/cudnn/lib:$CONDA_PREFIX/pkgs/cuda-nvcc-11.8.89-0/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/opt/software/conda/lib:$LD_LIBRARY_PATH
%test
	echo $PATH
%runscript
	exec "$@"
```

We can also build a container image out of a def file

- We create what is called a definition file, which will create the necessary instructions to build the container image

```
%environment
export CONDA_PREFIX='/opt/software/conda'
export PATH='/opt/software/conda/bin':$PATH
export LD_LIBRARY_PATH=$CONDA_PREFIX/lib/python3.10/site-packages/nvidia/cudnn/lib:$CONDA_PREFIX/pkgs/cuda-nvcc-11.8.89-0/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/opt/software/conda/lib:$LD_LIBRARY_PATH
%test
    echo $PATH
%runscript
    exec "$@"
```

It can be used to test the ENV at the end

e.g. if the binaries where downloaded properly or a software runs as expected in a testing ENV.

We can also build a container image out of a def file

- We create what is called a definition file, which will create the necessary instructions to build the container image

```
%environment
export CONDA_PREFIX='/opt/software/conda'
export PATH='/opt/software/conda/bin':$PATH
export LD_LIBRARY_PATH=$CONDA_PREFIX/lib/python3.10/site-
packages/nvidia/cudnn/lib:$CONDA_PREFIX/pkgs/cuda-nvcc-11.8.89-
0/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/opt/software/conda/lib:$LD_LIBRARY_PATH
%test
	echo $PATH
%runscript
	exec "$@"
```

Contents are written to a file when the container is running.

University of Colorado
Anschutz Medical Campus

More information here:
https://apptainer.org/docs/user/main/definition_files.html

We can also build a container image out of a def file

- We run the build command out of the definition file.

```
@c3cpu-a2-u32-2 scripts]$ apptainer build splice_conda.img splice_conda.def
```

- Warning! You can only build a container that way on Alpine with apptainer and not singularity!

More information here:
https://curc.readthedocs.io/en/latest/software/containerization.html

Let’s submit a container application slurm script

- We run the build command out of the definition file.

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --partition=amilan
#SBATCH --time=03:00:00
#SBATCH --mem=100G
#SBATCH --job-name="rsem_prepare_and_run”
#SBATCH -o "rsem_star.%j.out”
#SBATCH -e "rsem_star.%j.err”
#SBATCH --account=amc-general
#SBATCH --qos=normal
```

Always request 1 node, unless you are using a distributed application

Let’s submit a container application slurm script

- We run the build command out of the definition file.

```
export ALPINE_SCRATCH=/gpfs/alpine1/scratch/$USERexport
APPTAINER_TMPDIR=$ALPINE_SCRATCH/apptainer/tmpexport
APPTAINER_CACHEDIR=$ALPINE_SCRATCH/apptainer/cache
mkdir -pv $APPTAINER_CACHEDIR $APPTAINER_TMPD
```

Always export tmp related
Directory variables to your scratch!
In case you are using **apptainer!**

Let’s submit a container application slurm script

- We run the build command out of the definition file.

```
export ALPINE_SCRATCH=/gpfs/alpine1/scratch/$USERexport
export SINGULARITY_TMPDIR=$ALPINE_SCRATCH/singularity/tmp
export SINGULARITY_CACHEDIR=$ALPINE_SCRATCH/singularity/cache
mkdir -pv $SINGULARITY_CACHEDIR $SINGULARITY_TMPDIR
```

In case you use **singularity** instead!

Let’s submit a container application slurm script

- You might need to bind some key directories or filesystems that you will export inside your container when running it.

```
export CONTAIN_DIR=/scratch/alpine/kfotso@xsede.org/Seq
cd ${CONTAIN_DIR}
input_dir="${PWD}/data/trimmedReads”
output_dir="${PWD}/Seq_data/data/trimmedReads/rsem_results”
annotation_dir=/scratch/alpine/kfotso@xsede.org/Seq/annotation/hg38
rsem_ref_dir="$annotation_dir/rsem_ref”
star_genome_dir="$annotation_dir/star_genome”
log_dir="${PWD}/AnkleOA_RNA_Seq/blood_data/trimmedReads/logs"
```

Export directories

Let’s submit a container application slurm script

- You mind need to bind some key directories or filesystems that you will export inside your container when running it.

```
apptainer exec -H $CONTAIN_DIR --bind=$input_dir,$output_dir \
$CONTAIN_DIR/RNA_seq_updated_STAR.sif STAR –runMode genomeGenerate \
--quantMode TranscriptomeSAM \
--genomeDir "$star_genome_dir" \
--genomeFastaFiles "rsem_ref/GRCh38.primary_assembly.genome.fa" \
--sjdbGTFfile "gencode.v47.annotation.gtf" \
--sjdbOverhang 100 \
--runThreadN $SLURM_NTASKS
```

exec is a command that will directly run the container Image.

Let’s submit a container application slurm script

- You mind need to bind some key directories or filesystems that you will export inside your container when running it.

```
apptainer exec -H $CONTAIN_DIR --bind=$input_dir,$output_dir \
$CONTAIN_DIR/RNA_seq_updated_STAR.sif STAR –runMode genomeGenerate \
--quantMode TranscriptomeSAM \
--genomeDir "$star_genome_dir" \
--genomeFastaFiles "rsem_ref/GRCh38.primary_assembly.genome.fa" \
--sjdbGTFfile "gencode.v47.annotation.gtf" \
--sjdbOverhang 100 \
--runThreadN $SLURM_NTASKS
```

exec is a command that will directly run the container image.

This command works with singularity as well.

Let’s submit a container application slurm script

- You mind need to bind some key directories or filesystems that you will export inside your container when running it.

```
apptainer exec -H $CONTAIN_DIR --bind=$input_dir,$output_dir \
$CONTAIN_DIR/RNA_seq_updated_STAR.sif STAR –runMode genomeGenerate \
--quantMode TranscriptomeSAM \
--genomeDir "$star_genome_dir" \
--genomeFastaFiles "rsem_ref/GRCh38.primary_assembly.genome.fa" \
--sjdbGTFfile "gencode.v47.annotation.gtf" \
--sjdbOverhang 100 \
--runThreadN $SLURM_NTASKS
```

Specify a home directory as the $CONTAIN_DIR variable

Let’s submit a container application slurm script

- You mind need to bind some key directories or filesystems that you will export inside your container when running it.

```
apptainer exec -H $CONTAIN_DIR --bind=$input_dir,$output_dir \
$CONTAIN_DIR/RNA_seq_updated_STAR.sif STAR –runMode genomeGenerate \
--quantMode TranscriptomeSAM \
--genomeDir "$star_genome_dir" \
--genomeFastaFiles "rsem_ref/GRCh38.primary_assembly.genome.fa" \
--sjdbGTFfile "gencode.v47.annotation.gtf" \
--sjdbOverhang 100 \
--runThreadN $SLURM_NTASKS
```

We can bind additional directories and filesystems with --bind

Let’s submit a container application slurm script

- You mind need to bind some key directories or filesystems that you will export inside your container when running it.

```
apptainer exec -H $CONTAIN_DIR --bind=$input_dir,$output_dir \
$CONTAIN_DIR/RNA_seq_updated_STAR.sif STAR –runMode genomeGenerate \
--quantMode TranscriptomeSAM \
--genomeDir "$star_genome_dir" \
--genomeFastaFiles "rsem_ref/GRCh38.primary_assembly.genome.fa" \
--sjdbGTFfile "gencode.v47.annotation.gtf" \
--sjdbOverhang 100 \
--runThreadN $SLURM_NTASKS
```

Path to our STAR sif image where we call STAR.

Let’s submit a container application slurm script

- You mind need to bind some key directories or filesystems that you will export inside your container when running it.

```
apptainer exec -H $CONTAIN_DIR --bind=$input_dir,$output_dir \
$CONTAIN_DIR/RNA_seq_updated_STAR.sif STAR –runMode genomeGenerate \
--quantMode TranscriptomeSAM \
--genomeDir "$star_genome_dir" \
--genomeFastaFiles "rsem_ref/GRCh38.primary_assembly.genome.fa" \
--sjdbGTFfile "gencode.v47.annotation.gtf" \
--sjdbOverhang 100 \
--runThreadN $SLURM_NTASKS
```

Note that we can call our number of tasks with $SLURM_NTASKS.

For #SBATCH --cpus-per-task, you may use

$SLURM_CPUS_PER_TASK

How to run singularity on a gpu partition?

- Here, we take an example using a deepvariant container image on Alpine from here: https://github.com/google/deepvariant/blob/r1.6.1/docs/deepvariant-quick-start.md

How to run singularity on a gpu partition?

- Here, we take an example using a deepvariant container image on Alpine from here: https://github.com/google/deepvariant/blob/r1.6.1/docs/deepvariant-quick-start.md

```
#!/bin/bash
#SBATCH --job-name=deepvariant_gpu
#SBATCH --nodes=1
#SBATCH --ntasks=16 # This is to select the amount of cores for the part of your code that will run on the CPU
#SBATCH --time=24:00:00  # This is to change the walltime
#SBATCH --partition=aa100 # Name of the NVIDIA gpu partition
#SBATCH --gres=gpu:1 # Requesting 1 GPU from the GPU node (There is a total of 3 gpus per node)
#SBATCH --constraint=gpu80
#SBATCH --output=deepvariant_gpu-%j.out
#SBATCH --error=deepvariant_gpu-%j.err
#SBATCH --qos=normal
```

How to run singularity on a gpu partition?

- Here, we take an example using a deepvariant container image on Alpine from here: https://github.com/google/deepvariant/blob/r1.6.1/docs/deepvariant-quick-start.md

```
#!/bin/bash
#SBATCH --job-name=deepvariant_gpu
#SBATCH --nodes=1
#SBATCH --ntasks=16 # This is to select the amount of cores for the part of your code that will run on the CPU
#SBATCH --time=24:00:00  # This is to change the walltime
#SBATCH --partition=aa100 # Name of the NVIDIA gpu partition
#SBATCH --gres=gpu:1 # Requesting 1 GPU from the GPU node (There is a total of 3 gpus per node)
#SBATCH --constraint=gpu80 # This will select only gpu nodes with 80G VRAM per gpu.
#SBATCH --output=deepvariant_gpu-%j.out
#SBATCH --error=deepvariant_gpu-%j.err
#SBATCH --qos=normal
```

How to run singularity on a gpu partition?

- Here, we take an example using a deepvariant container image on Alpine from here: https://github.com/google/deepvariant/blob/r1.6.1/docs/deepvariant-quick-start.md
- In this case, there are some system environment variable and paths that we needed to export. We call it myenvs:

```
LD_LIBRARY_PATH=/usr/local/lib/python3.8/dist-packages/nvidia/cublas/lib:/usr/local/nvidia/lib:/usr/local/nvidia/lib64:/.singularity.d/libs
TMPDIR=/gpfs/alpine1/scratch/kfotso@xsede.org/mytmp
```

How to run singularity on a gpu partition?

- Here, we take an example using a deepvariant container image on Alpine from here: https://github.com/google/deepvariant/blob/r1.6.1/docs/deepvariant-quick-start.md
- Then , we can run our container, assuming that we have exported all the tmp variables and defined all the directory variables to bind as well.

```
apptainer exec -H $CONTAIN_DIR --
bind=$CONTAIN_DIR,$ALPINE_SCRATCH,$RESULT_DIR,$LONG_READ_DIR,$TMPDIR

--nv

--env-file myenvs \ deepvariant-gpu-1_6_1_v2.sif /opt/deepvariant/bin/run_deepvariant \

--model_type=MODEL --ref=$REF --reads=$READS \

--output_vcf=$RESULT_DIR/mydata.vcf.gz \

--intermediate_results_dir $ALPINE_SCRATCH/intermediate_results_dir

--sample_name=SAMPLE --num_shards=$SLURM_NTASKS
```

How to run singularity on a gpu partition?

- Here, we take an example using a deepvariant container image on Alpine from here: https://github.com/google/deepvariant/blob/r1.6.1/docs/deepvariant-quick-start.md
- Then , we can run our container, assuming that we have exported all the tmp variables and defined all the directory variables to bind as well.

```
apptainer exec -H $CONTAIN_DIR --
bind=$CONTAIN_DIR,$ALPINE_SCRATCH,$RESULT_DIR,$LONG_READ_DIR,$TMPDIR

--nv

--env-file myenvs \

deepvariant-gpu-1_6_1_v2.sif /opt/deepvariant/bin/run_deepvariant \

--model_type=MODEL--ref=$REF --reads=$READS \

--output_vcf=$RESULT_DIR/mydata.vcf.gz \

--intermediate_results_dir $ALPINE_SCRATCH/intermediate_results_dir

--sample_name=SAMPLE --num_shards=$SLURM_NTASKS
```

--env-file allows to call directly those additional system variable directories into the container image, when running it.

How to run singularity on a gpu partition?

- Here, we take an example using a deepvariant container image on Alpine from here: https://github.com/google/deepvariant/blob/r1.6.1/docs/deepvariant-quick-start.md
- Then , we can run our container, assuming that we have exported all the tmp variables and defined all the directory variables to bind as well.

```
apptainer exec -H $CONTAIN_DIR --
bind=$CONTAIN_DIR,$ALPINE_SCRATCH,$RESULT_DIR,$LONG_READ_DIR,$TMPDIR
--nv
--env-file myenvs \
deepvariant-gpu-1_6_1_v2.sif /opt/deepvariant/bin/run_deepvariant \
--model_type=MODEL--ref=$REF --reads=$READS \
--output_vcf=$RESULT_DIR/mydata.vcf.gz \
--intermediate_results_dir $ALPINE_SCRATCH/intermediate_results_dir
--sample_name=SAMPLE--num_shards=$SLURM_NTASKS
```

--nv allows to mount nvidia cuda related libraries from your host env into your container env.

How to modify/edit a container image on Alpine?

- In this case, we modify a container image that runs on an AMD gpu on Alpine.

Use the commands below to access respectively an AMD gpu debug node and and AMD batch node.

sinteractive ***--partition=atesting_mi100*** *--qos=testing --time=01:00:00 --gres=gpu:1 --ntasks=4*

sinteractive ***--partition=ami100*** *--nodes=1 --ntasks=4 --gres=gpu:1 --time=00:10:00 –reservation=amc_workshop*

How to modify/edit a container image on Alpine?

- To build the container, we have to create an overlay. An overlay is a filesystem that is going to be mounted on top of the container and serve as a buffer to edit the container image.

- If you modify a .sif or .simg container image via an overlay, you will always need to call it from now on, as it retains all the edits.

How to modify/edit a container image on Alpine?

- To build the container, we have to create an overlay. An overlay is a filesystem that is going to be mounted on top of the container and serve as a buffer to edit the container image.

- If you modify a .sif or .simg container image via an overlay, you will always need to call it from now on, as it retains all the edits.

- In this case, we try to build and modify and AMD gpu container.

apptainer build dev-ubuntu-20.04_latest.sif docker://rocm/dev-ubuntu-20.04_latest

How to modify/edit a container image on Alpine?

- To build the container, we have to create an overlay. An overlay is a filesystem that is going to be mounted on top of the container and serve as a buffer to edit the container image.

- If you modify a .sif or .simg container image via an overlay, you will always need to call it from now on, as it retains all the edits.

- In this case, we try to build and modify and AMD gpu container.

```
apptainer build dev-ubuntu-20.04_latest.sif docker://rocm/dev-ubuntu-20.04_latest
```

```
apptainer overlay create --fakeroot --sparse --size 100000 sparse_simple_overlay.img
```

How to modify/edit a container image on Alpine?

- To build the container, we have to create an overlay. An overlay is a filesystem that is going to be mounted on top of the container and serve as a buffer to edit the container image.

- If you modify a .sif or .simg container image via an overlay, you will always need to call it from now on, as it retains all the edits.

- In this case, we try to build and modify and AMD gpu container.

```
apptainer build dev-ubuntu-20.04_latest.sif docker://rocm/dev-ubuntu-20.04_latest
```

```
apptainer overlay create --fakeroot --sparse --size 100000 sparse_simple_overlay.img
```

Sparse overlay only takes up space on disk as data is written over it.

How to modify/edit a container image on Alpine?

- To build the container, we have to create an overlay. An overlay is a filesystem that is going to be mounted on top of the container and serve as a buffer to edit the container image.

- If you modify a .sif or .simg container image via an overlay, you will always need to call it from now on, as it retains all the edits.

- In this case, we try to build and modify and AMD gpu container.

```
apptainer build dev-ubuntu-20.04_latest.sif docker://rocm/dev-ubuntu-20.04_latest
```

```
apptainer overlay create --fakeroot --sparse --size 100000 sparse_simple_overlay.img
```

100000 MiB size

How to modify/edit a container image on Alpine?

- To build the container, we have to create an overlay. An overlay is a filesystem that is going to be mounted on top of the container and serve as a buffer to edit the container image.

- If you modify a .sif or .simg container image via an overlay, you will always need to call it from now on, as it retains all the edits.

- In this case, we try to build and modify and AMD gpu container.

```
apptainer build dev-ubuntu-20.04_latest.sif docker://rocm/dev-ubuntu-20.04_latest
```

```
apptainer overlay create --fakeroot --sparse --size 100000 sparse_simple_overlay.img
```

- Allows user to build root without sudo priviledges and interact with the overlay

How to modify/edit a container image on Alpine?

```
echo "********************************* Copying the .sif and .img files  *************************"
cp /scratch/alpine/kfotso@xsede.org/test_container_docker/sparse_3_simple_overlay.img .
cp /scratch/alpine/kfotso@xsede.org/test_container_docker/build-dev-ubuntu-20.04_latest.sif .
export CONTAIN_DIR=${PWD}

echo "********************************* Running the built container *************************"
# --containall is very important as it allows to transfer video permission to the container
# Otherwise you will not see the AMD GPU
#--overlay allows to create a filesystem on top of the immutable .sif container
# so that you can modify it at will with the --fakeroot option
# To install packages and run pipeline inside the container we run:
apptainer shell -H $CONTAIN_DIR --bind=/dev/kfd,/dev/dri --fakeroot --rocm --containall --overlay sparse_3_simple_overlay.img build-dev-ubuntu-20.04_
```

--rocm is the equivalent of –nv for nvidia and allows to import system related AMD gpu libraries from the host into the container. call

How to modify/edit a container image on Alpine?

```
echo "********************************* Copying the .sif and .img files  *************************"
cp /scratch/alpine/kfotso@xsede.org/test_container_docker/sparse_3_simple_overlay.img .
cp /scratch/alpine/kfotso@xsede.org/test_container_docker/build-dev-ubuntu-20.04_latest.sif .
export CONTAIN_DIR=${PWD}

echo "********************************* Running the built container *************************"
# --containall is very important as it allows to transfer video permission to the container
# Otherwise you will not see the AMD GPU
#--overlay allows to create a filesystem on top of the immutable .sif container
# so that you can modify it at will with the --fakeroot option
# To install packages and run pipeline inside the container we run:
apptainer shell -H $CONTAIN_DIR --bind=/dev/kfd,/dev/dri --fakeroot --rocm --containall --overlay sparse_3_simple_overlay.img build-dev-ubuntu-20.04_
```

With shell, one can modify the container interactively and thus install packages, test software etc …

How to modify/edit a container image on Alpine?

```
echo "********************************* Copying the .sif and .img files  *************************"
cp /scratch/alpine/kfotso@xsede.org/test_container_docker/sparse_3_simple_overlay.img .
cp /scratch/alpine/kfotso@xsede.org/test_container_docker/build-dev-ubuntu-20.04_latest.sif .
export CONTAIN_DIR=${PWD}

echo "********************************* Running the built container *************************"
# --containall is very important as it allows to transfer video permission to the container
# Otherwise you will not see the AMD GPU
#--overlay allows to create a filesystem on top of the immutable .sif container
# so that you can modify it at will with the --fakeroot option
# To install packages and run pipeline inside the container we run:
apptainer shell -H $CONTAIN_DIR --bind=/dev/kfd,/dev/dri --fakeroot --rocm --containall --overlay sparse_3_simple_overlay.img build-dev-ubuntu-20.04_
```

We export important filesystems necessary to run an AMD gpu inside a container.

How to modify/edit a container image on Alpine?

- Inside the container image, we copy a python script to /usr/bin to run it.

```
[kfotso@xsede.org@c3gpu-c2-u17 test_container_docker]$ apptainer shell -H $CONTAIN_DIR --bind=/dev/kfd,/dev/dri --fakeroot --rocm --containall --overlay spars
e_3_simple_overlay.img build-dev-ubuntu-20.04_latest.sif
]INFO:    User not listed in /etc/subuid, trying root-mapped namespace
INFO:    Using fakeroot command combined with root-mapped namespace
WARNING: Skipping /dev/kfd bind mount: already mounted
WARNING: Skipping /dev/dri bind mount: already mounted
INFO:    unknown argument ignored: lazytime
Apptainer> ]     ds
Apptainer>
Apptainer> cp tensors.py /usr/bin/
```

We are inside the apptainer shell.

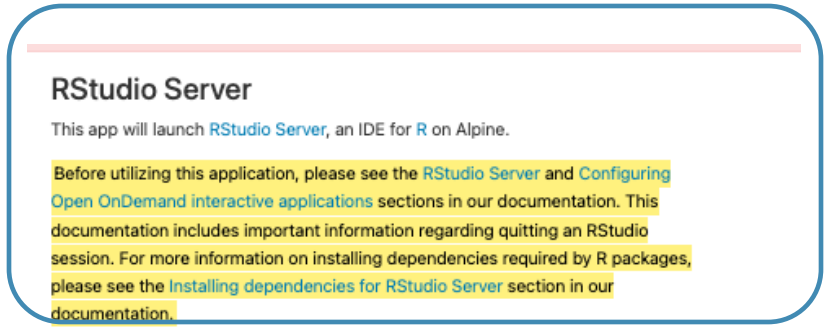
Rstudio on Alpine

What is Ondemand Rstudio?

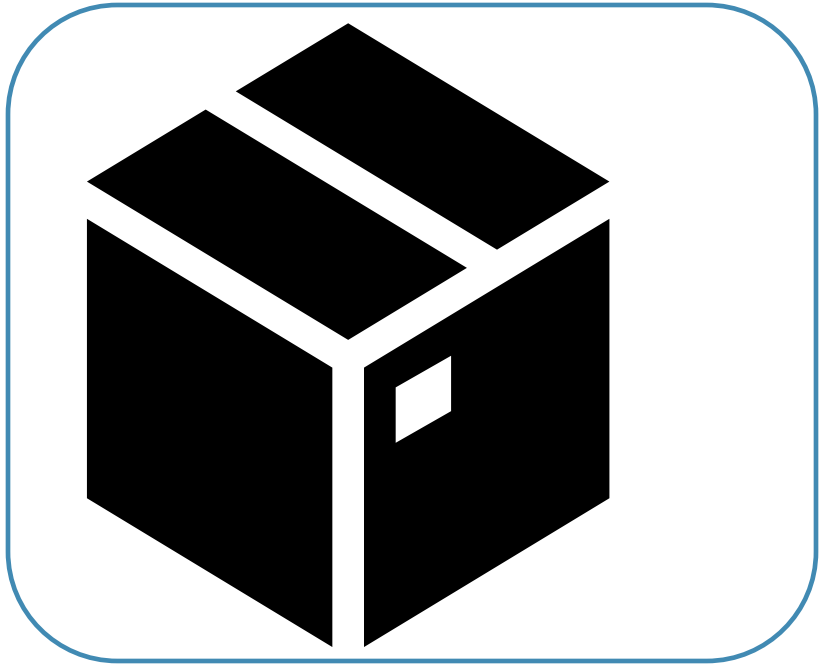
- It was built on top of an apptainer container image, thus, it runs on a different operating system than Alpine.

- A container is a tool that allows you to run different applications that were built on different operating systems on Alpine.

- The Rstudio .sif image is located at */curc/sw/containers/open_ondemand/rstudio-server-4.4.1.sif*



Rstudio



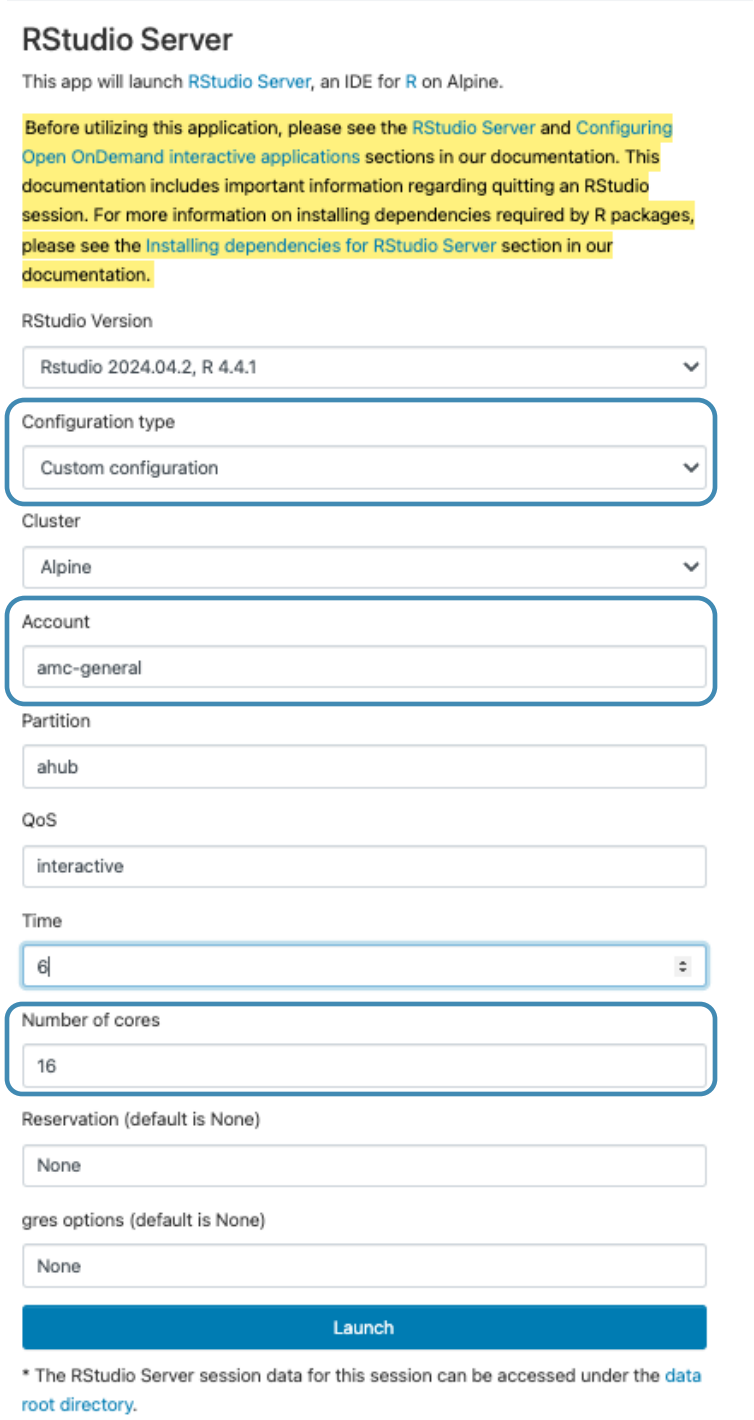
Container

Rstudio on Alpine

Use Rstudio on Alpine if:

- If you are used to Rstudio outside of Alpine
- You will need access to a GUI.
- If you think that most of your pipelines will not require more than 16 cores or 60GB of RAM.
- **Note: *** If you are new to Rstudio on Alpine, please refer to this guide: https://github.com/kf-cuanschutz/CU-Anschutz-HPC-documentation/tree/main/Rstudio_related_scripts

* If you wish to run your Rstudio contained environment as a slurm batch script, please refer to this guide: https://github.com/kf-cuanschutz/CU-Anschutz-HPC-documentation/blob/main/Rstudio_on_Slurm_.md



University of Colorado
Anschutz Medical Campus

Container orchestration

- An easy way to orchestrate if you are running a container image with multiple similar like files is to use job arrays.

Container orchestration

- An easy way to orchestrate if you are running a container image with multiple similar like files is to use job arrays.

#!/bin/bash

#SBATCH --nodes=1

#SBATCH --ntasks=10 # Number of CPU cores. As you request more CPU cores, you are also getting more CPU memory. ***You have about 3.8G per core***

#SBATCH --time=03:00:00 # Walltime

#SBATCH --partition=aa100 # This is the name of the NVIDIA GPU partition. Made of nodes containing 3x A100 gpus.

#SBATCH --gres=gpu:1 # Here we are requesting 1 gpu
#SBATCH --job-name=cellbender_gpu # Name of the job that will be submitted.

Container orchestration

- An easy way to orchestrate if you are running a container image with multiple similar like files is to use job arrays.

#!/bin/bash

#SBATCH --nodes=1

#SBATCH --ntasks=10 # Number of CPU cores. As you request more CPU cores, you are also getting more CPU memory. ***You have about 3.8G per core***

#SBATCH --time=03:00:00 # Walltime

#SBATCH --partition=aa100 # This is the name of the NVIDIA GPU partition. Made of nodes containing 3x A100 gpus.

#SBATCH --gres=gpu:1 # Here we are requesting 1 gpu
#SBATCH --job-name=cellbender_gpu # Name of the job that will be submitted.

#SBATCH --array=1-72

We add *add into the slurm header which correspond to the number of samples.*

Container orchestration

- An easy way to orchestrate if you are running a container image with multiple similar like files is to use job arrays.

```
export IndexID=$SLURM_ARRAY_TASK_ID
```

Then, we fetch the array task index corresponding to the slurm array subjobID:

Container orchestration

- An easy way to orchestrate if you are running a container image with multiple similar like files is to use job arrays.

```
export IndexID=$SLURM_ARRAY_TASK_ID
export SAMPLE="RESULT_${IndexID}"
mkdir $SAMPLE
export Data_dir=/pl/active/foolab/shared/$SAMPLE
```

We assign it to a variable called SAMPLE

```
apptainer exec -H $CONTAIN_DIR --
bind=$Data_dir,$ALPINE_SCRATCH,$SAMPLE
--nv \
cellbender_gpu.sif cellbender remove-background \
--cuda  --fpr 0.1  --input $Data_dir/raw_feature_bc_matrix.h5 \
--output $SAMPLE/output.h5
```

Container orchestration

- An easy way to orchestrate if you are running a container image with multiple similar like files is to use job arrays.

```
export IndexID=$SLURM_ARRAY_TASK_ID

export SAMPLE="RESULT_${IndexID}"

mkdir $SAMPLE

export Data_dir=/pl/active/foolab/shared/$SAMPLE
```

Exporting data where the sample ID is located

```
apptainer exec -H $CONTAIN_DIR --
bind=$Data_dir,$ALPINE_SCRATCH,$SAMPLE

--nv \

cellbender_gpu.sif cellbender remove-background \

--cuda  --fpr 0.1  --input $Data_dir/raw_feature_bc_matrix.h5 \
--output $SAMPLE/output.h5
```

Container orchestration

- An easy way to orchestrate if you are running a container image with multiple similar like files is to use job arrays.

```
export IndexID=$SLURM_ARRAY_TASK_ID

export SAMPLE="RESULT_${IndexID}"

mkdir $SAMPLE

export Data_dir=/pl/active/foolab/shared/$SAMPLE


apptainer exec -H $CONTAIN_DIR --
bind=$Data_dir,$ALPINE_SCRATCH,$SAMPLE

--nv \

cellbender_gpu.sif cellbender remove-background \

--cuda  --fpr 0.1  --input $Data_dir/raw_feature_bc_matrix.h5 \
--output $SAMPLE/output.h5
```

Exporting the array ID properly into the container image


University of Colorado Anschutz Medical Campus
THANK YOU