Overview of GPU usage on Alpine

Kevin Fotso

GPU nodes on Alpine

12 NVIDIA GPU NODES (3 GPU EACH)

8 AMD GPU NODES (3 GPU EACH)

GPU partition on Alpine

AA100 -> NVIDIA RELATED PARTITION; MAX WALLTIME 24 HOURS

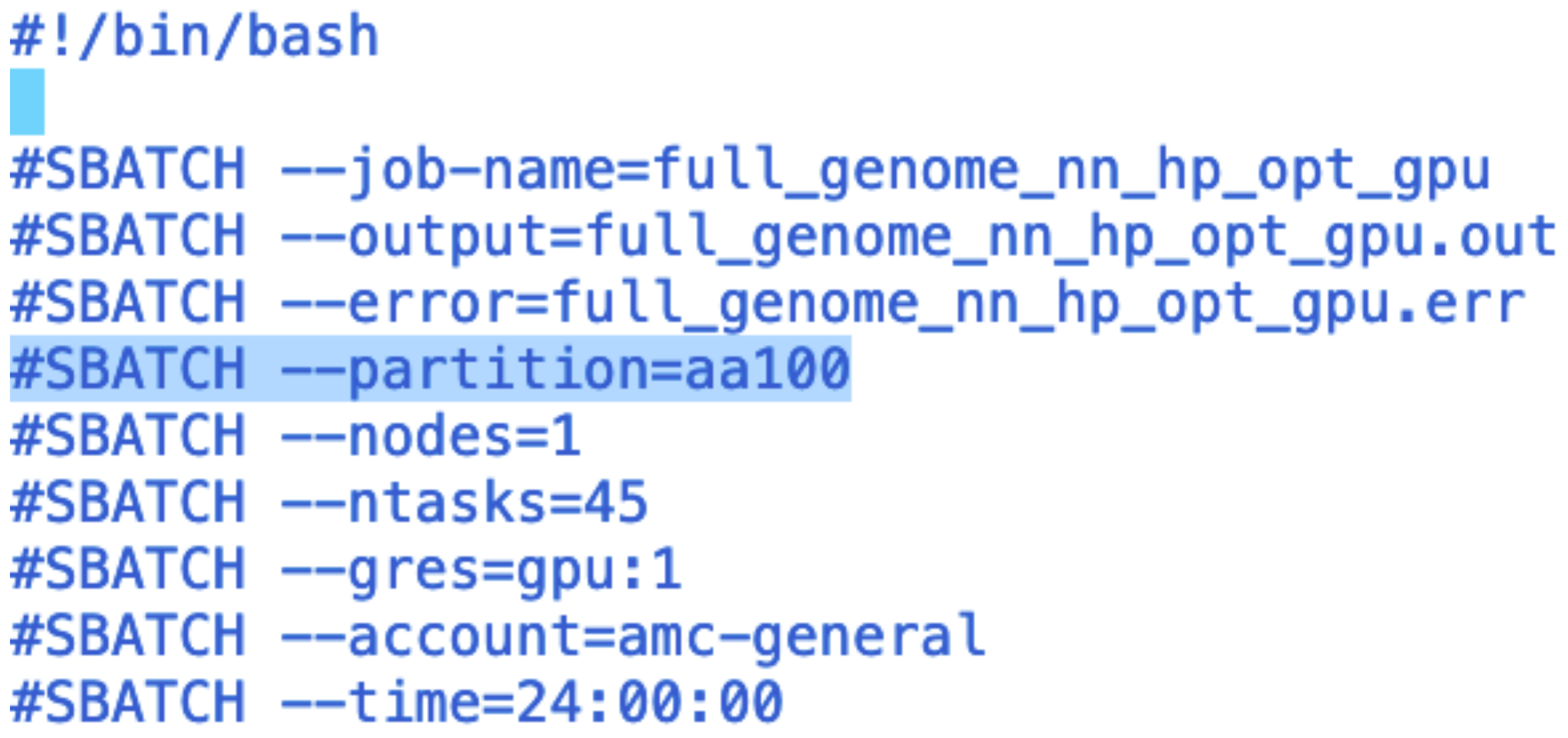
AMI100 -> AMD RELATED PARTITION; MAX WALLTIME 24 HOURS

ATESTING_A100 -> NVIDIA TESTING PARTITION; MAX WALLTIME 1 HOUR

ATESTING_AMI100 -> AMD TESTING PARTITION; MAX WALLTIME 1 HOUR

IMPORTANT: TO REQUEST MORE THAN 24 HOURS WALLTIME USE --QOS=LONG

Access to NVIDIA Job partition



```
#!/bin/bash

#SBATCH --job-name=full_genome_nn_hp_opt_gpu
#SBATCH --output=full_genome_nn_hp_opt_gpu.out
#SBATCH --error=full_genome_nn_hp_opt_gpu.err
#SBATCH --partition=aa100
#SBATCH --nodes=1
#SBATCH --ntasks=45
#SBATCH --gres=gpu:1
#SBATCH --account=amc-general
#SBATCH --time=24:00:00
```

- To access an AMD partition add: **--partition=ami100**
- Here, the number of gpus are allocated with: **--gres=gpu:1**

Watch out for cache size

- /home or /temp have a very small size. Please include the following in your slurm script.

```
export TMP=/scratch/alpine/$USER
export TEMP=/scratch/alpine/$USER
export TMPDIR=/scratch/alpine/$USER
export TEMPDIR=/scratch/alpine/$USER
```

Access to gpu nodes testing partitions

- [NVIDIA] sinteractive --partition=atesting_a100 --qos=testing --time=00:05:00 --gres=gpu:1 --ntasks=2
- [AMD] sinteractive --partition=atesting_mi100 --qos=testing --time=00:05:00 --gres=gpu:1 --ntasks=2

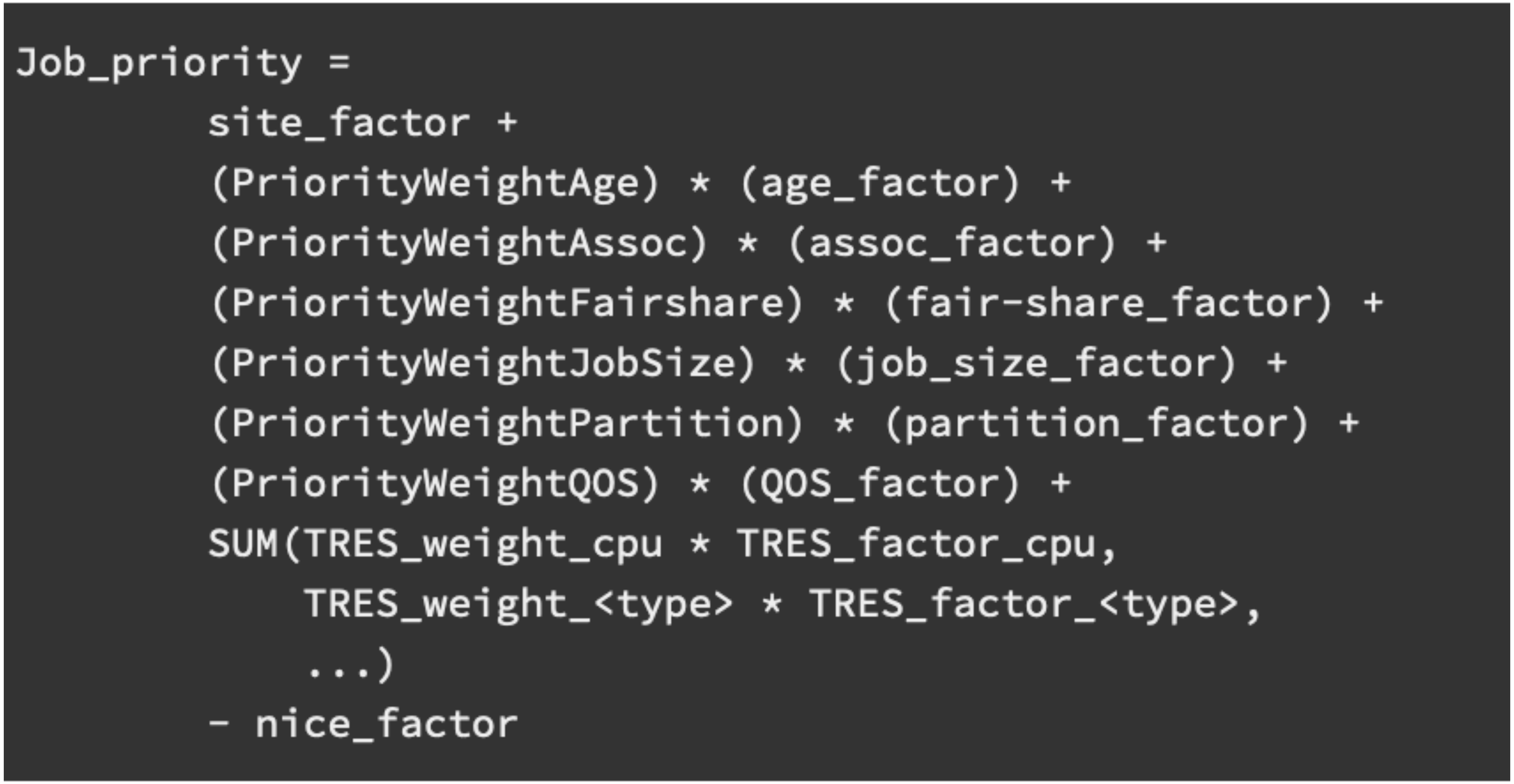
Slurm Quality of service (qos)

- Used to modify or constrain characteristics that a job can have.
- **--qos=normal** corresponds to a walltime of 24 hours and is the default.
- **--qos=long** corresponds to a walltime of up to 7 days
- **--qos=mem** corresponds to high memory jobs only (up to 1TB)

Fairshare overview

- Difference between the portion of computing resource that has been promised and the amount of resources that has been consumed.
- Level fairshare of 1 indicates average priority compared to other users in that account (amc-general)
- **module load slurmtools; levelfs $USER**

Job priority calculation formula



```
Job_priority =
        site_factor +
        (PriorityWeightAge) * (age_factor) +
        (PriorityWeightAssoc) * (assoc_factor) +
        (PriorityWeightFairshare) * (fair-share_factor) +
        (PriorityWeightJobSize) * (job_size_factor) +
        (PriorityWeightPartition) * (partition_factor) +
        (PriorityWeightQOS) * (QOS_factor) +
        SUM(TRES_weight_cpu * TRES_factor_cpu,
            TRES_weight_<type> * TRES_factor_<type>,
            ...)
        - nice_factor
```

Package availability (1)

Some packages that have been built and accessible through lmod.

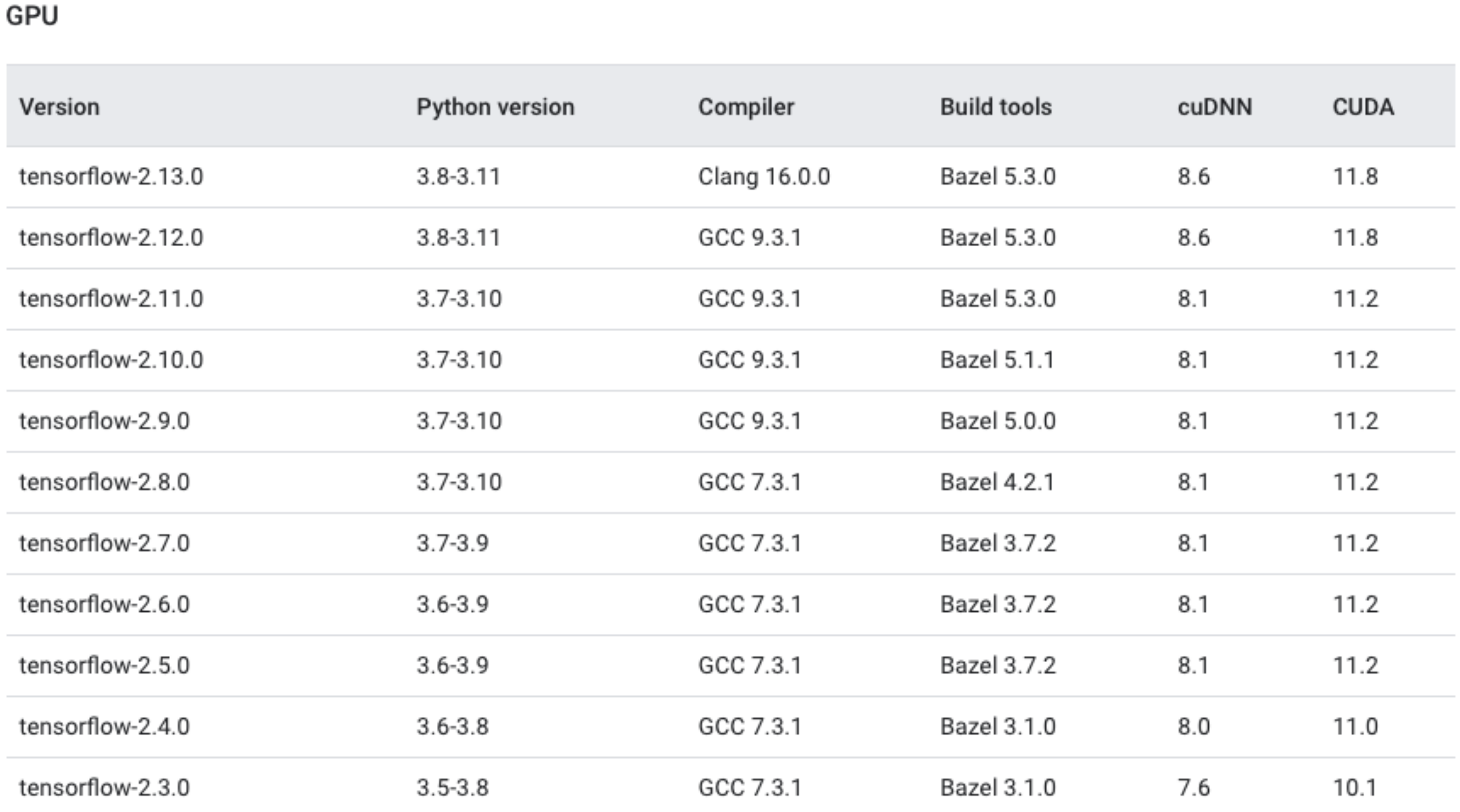
Adding new packages through lmod takes a lot of round of approval so it is recommended to build them locally.

Solutions: (cmake+make), Anaconda, pip, containers, spack etc ...

Submit a ticket at rc-help@Colorado.edu so that I can build it for you locally.

Package availability for ML (2)

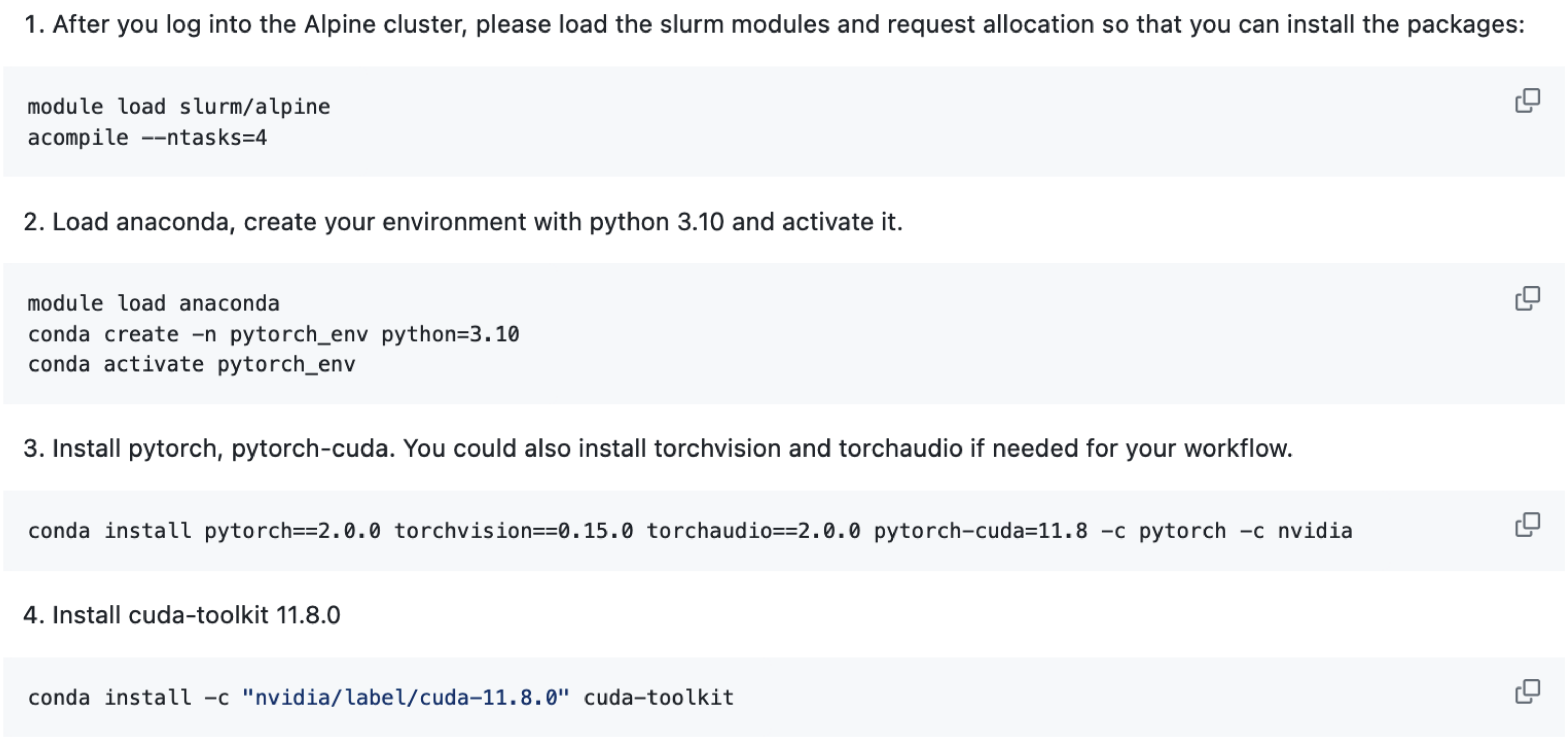
- Cuda 11.2, Cuda 11.3 and Cuda 11.4 on Alpine.
- Only cudnn 8.1 and 8.2 on Alpine.
- Can be problematic for DL build with GPU compatibility



GPU

Version	Python version	Compiler	Build tools	cuDNN	CUDA
tensorflow-2.13.0	3.8-3.11	Clang 16.0.0	Bazel 5.3.0	8.6	11.8
tensorflow-2.12.0	3.8-3.11	GCC 9.3.1	Bazel 5.3.0	8.6	11.8
tensorflow-2.11.0	3.7-3.10	GCC 9.3.1	Bazel 5.3.0	8.1	11.2
tensorflow-2.10.0	3.7-3.10	GCC 9.3.1	Bazel 5.1.1	8.1	11.2
tensorflow-2.9.0	3.7-3.10	GCC 9.3.1	Bazel 5.0.0	8.1	11.2
tensorflow-2.8.0	3.7-3.10	GCC 7.3.1	Bazel 4.2.1	8.1	11.2
tensorflow-2.7.0	3.7-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.6.0	3.6-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.5.0	3.6-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.4.0	3.6-3.8	GCC 7.3.1	Bazel 3.1.0	8.0	11.0
tensorflow-2.3.0	3.5-3.8	GCC 7.3.1	Bazel 3.1.0	7.6	10.1

Pytorch installation on Alpine [NVIDIA] (1)



1. After you log into the Alpine cluster, please load the slurm modules and request allocation so that you can install the packages:

```
module load slurm/alpine
acompile --ntasks=4
```

2. Load anaconda, create your environment with python 3.10 and activate it.

```
module load anaconda
conda create -n pytorch_env python=3.10
conda activate pytorch_env
```

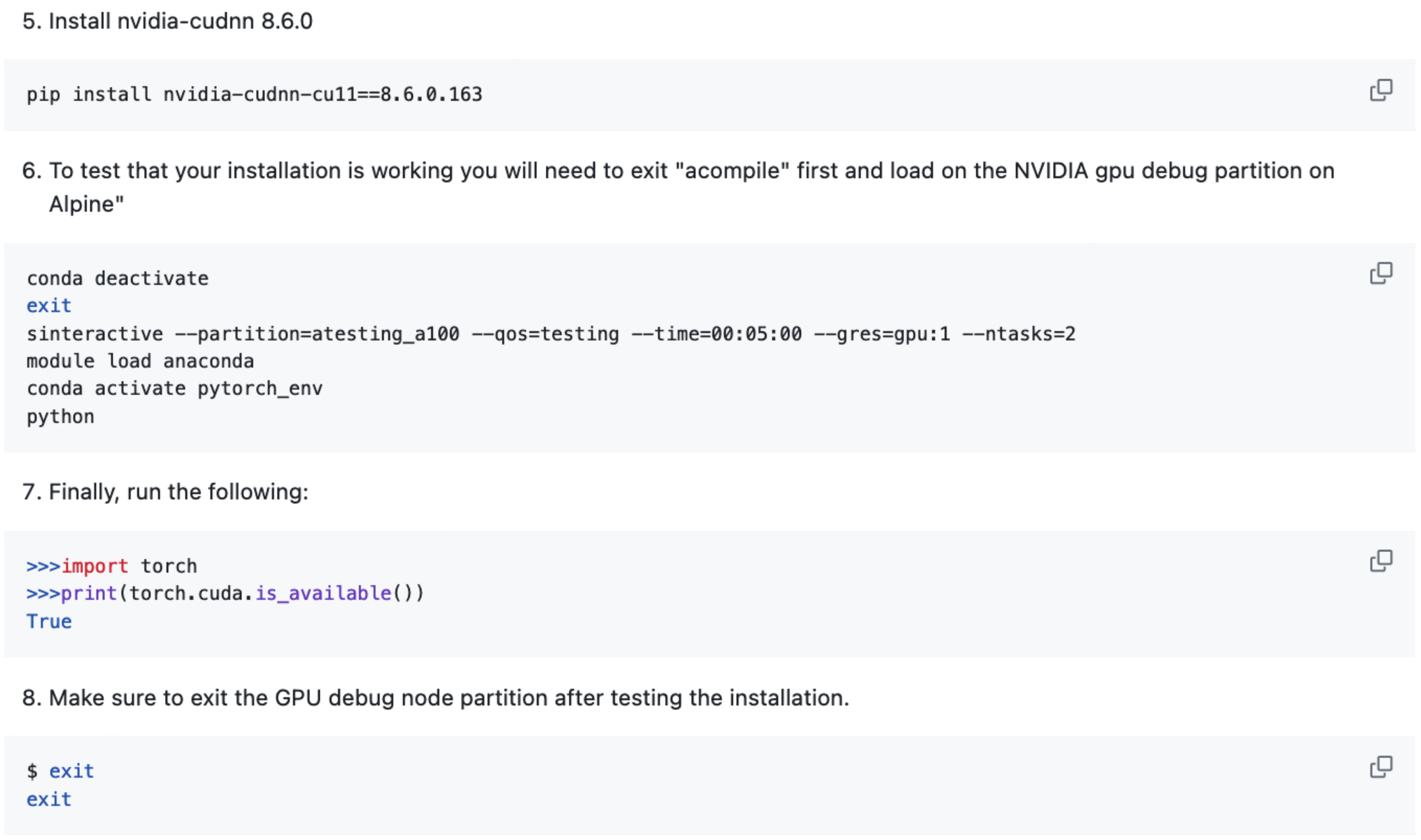
3. Install pytorch, pytorch-cuda. You could also install torchvision and torchaudio if needed for your workflow.

```
conda install pytorch==2.0.0 torchvision==0.15.0 torchaudio==2.0.0 pytorch-cuda=11.8 -c pytorch -c nvidia
```

4. Install cuda-toolkit 11.8.0

```
conda install -c "nvidia/label/cuda-11.8.0" cuda-toolkit
```

Pytorch installation on Alpine [NVIDIA] (2)



5. Install nvidia-cudnn 8.6.0

```
pip install nvidia-cudnn-cu11==8.6.0.163
```

6. To test that your installation is working you will need to exit "acompile" first and load on the NVIDIA gpu debug partition on Alpine"

```
conda deactivate
exit
sinteractive --partition=atesting_a100 --qos=testing --time=00:05:00 --gres=gpu:1 --ntasks=2
module load anaconda
conda activate pytorch_env
python
```

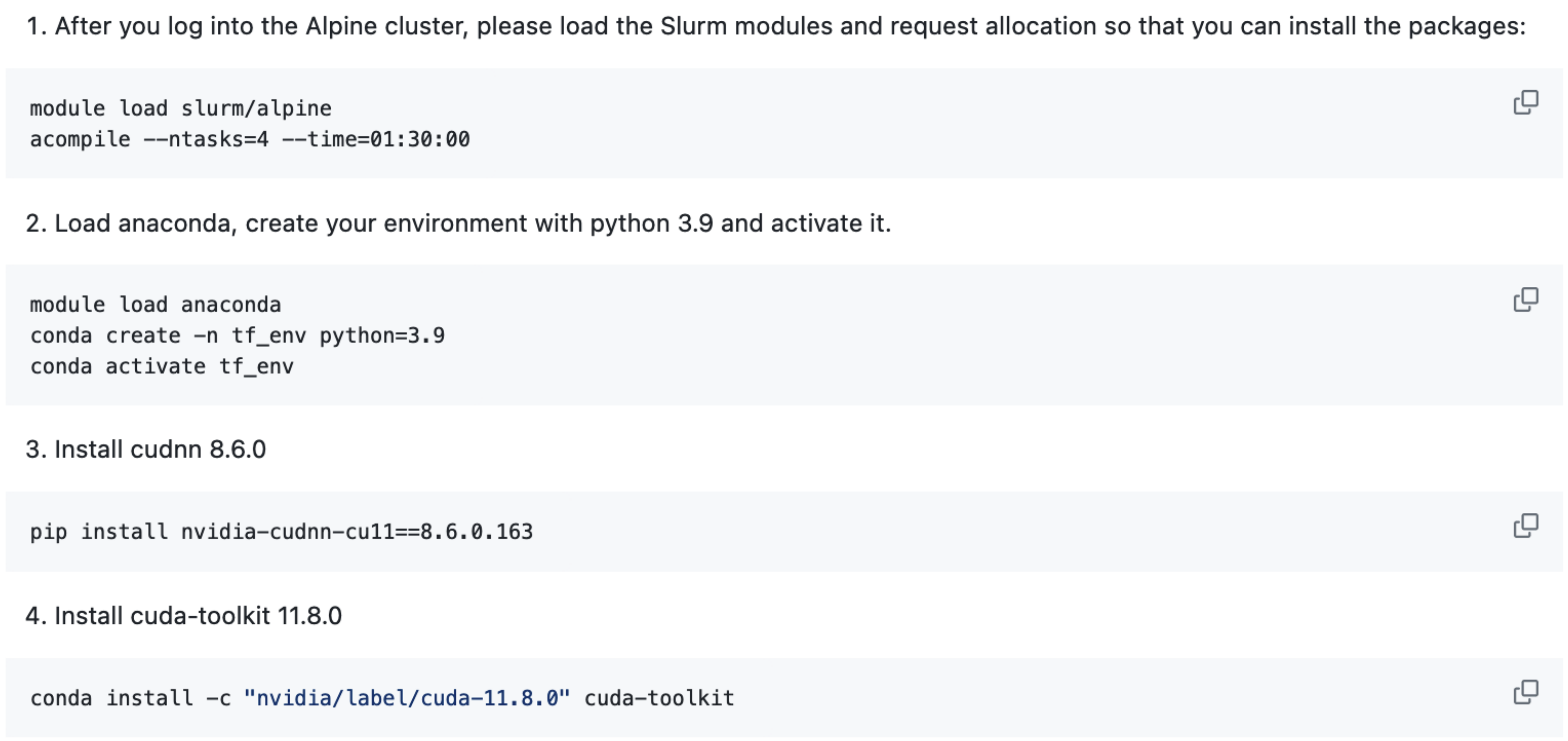
7. Finally, run the following:

```
>>>import torch
>>>print(torch.cuda.is_available())
True
```

8. Make sure to exit the GPU debug node partition after testing the installation.

```
$ exit
exit
```

Tensorflow installation on Alpine [NVIDIA] (1)



1. After you log into the Alpine cluster, please load the Slurm modules and request allocation so that you can install the packages:

```
module load slurm/alpine
acompile --ntasks=4 --time=01:30:00
```

2. Load anaconda, create your environment with python 3.9 and activate it.

```
module load anaconda
conda create -n tf_env python=3.9
conda activate tf_env
```

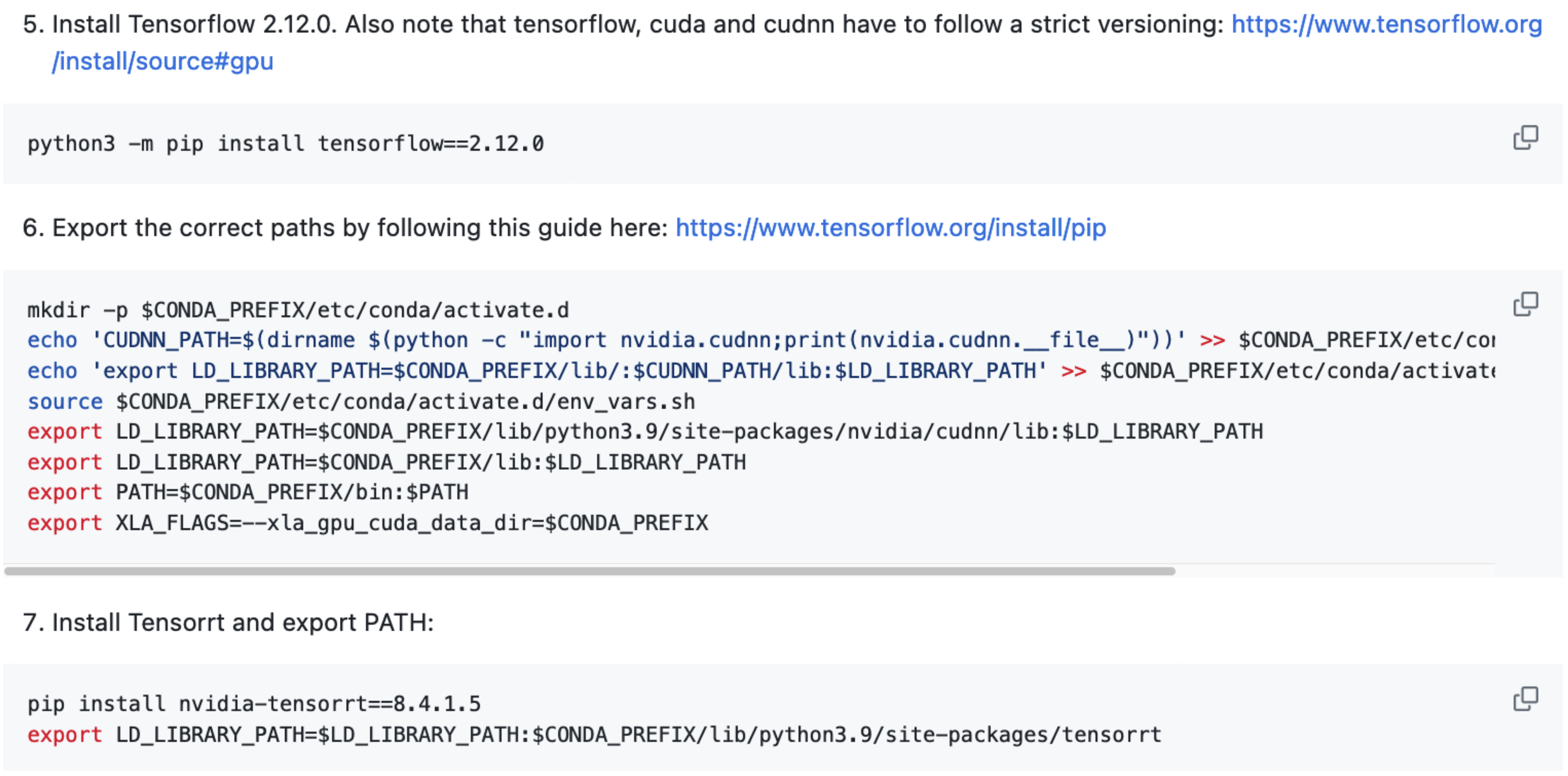
3. Install cudnn 8.6.0

```
pip install nvidia-cudnn-cu11==8.6.0.163
```

4. Install cuda-toolkit 11.8.0

```
conda install -c "nvidia/label/cuda-11.8.0" cuda-toolkit
```

Tensorflow installation on Alpine [NVIDIA] (2)



5. Install Tensorflow 2.12.0. Also note that tensorflow, cuda and cudnn have to follow a strict versioning: https://www.tensorflow.org/install/source#gpu

```
python3 -m pip install tensorflow==2.12.0
```

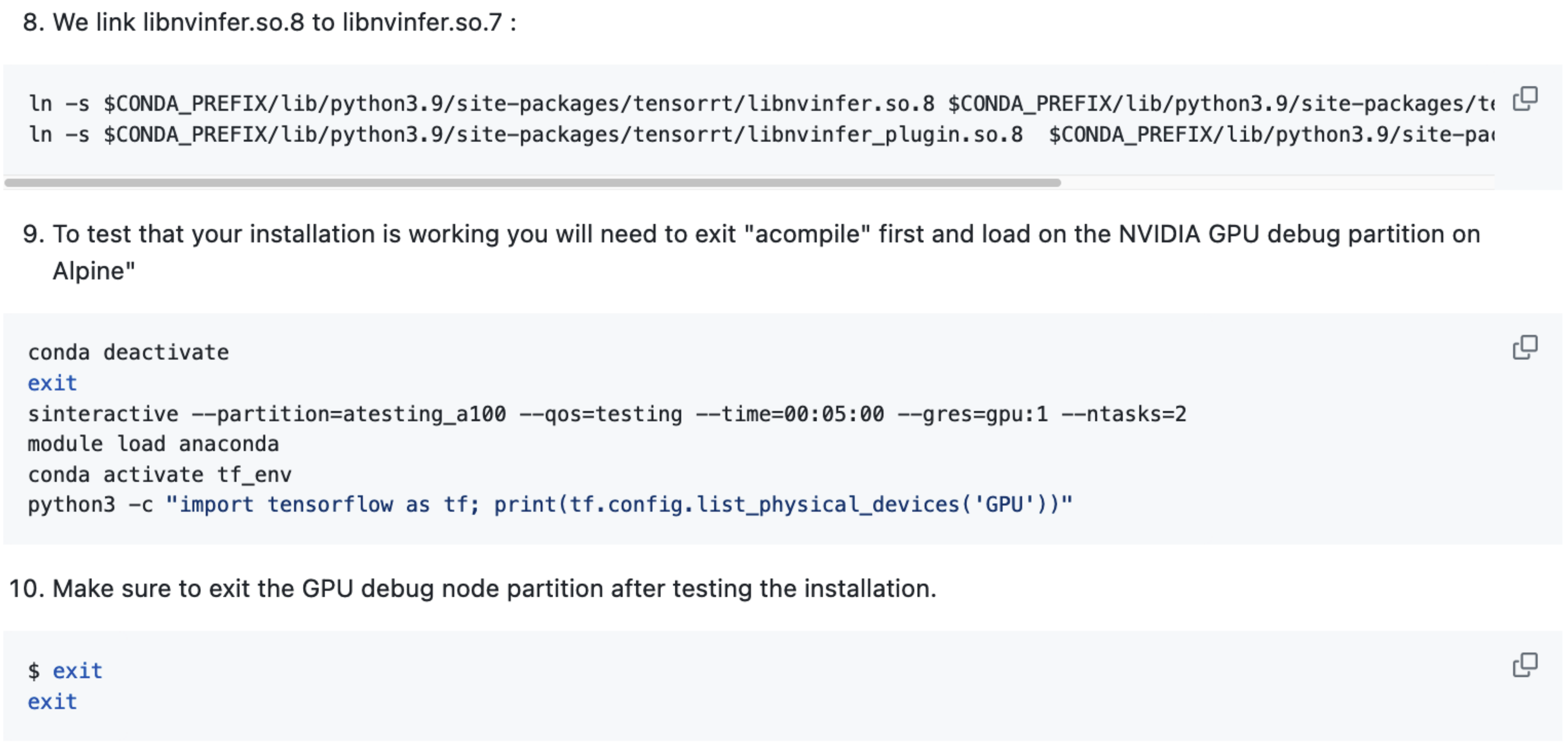
6. Export the correct paths by following this guide here: https://www.tensorflow.org/install/pip

```
mkdir -p $CONDA_PREFIX/etc/conda/activate.d
echo 'CUDNN_PATH=$(dirname $(python -c "import nvidia.cudnn;print(nvidia.cudnn.__file__)"))' >> $CONDA_PREFIX/etc/co
echo 'export LD_LIBRARY_PATH=$CONDA_PREFIX/lib/:$CUDNN_PATH/lib:$LD_LIBRARY_PATH' >> $CONDA_PREFIX/etc/conda/activat
source $CONDA_PREFIX/etc/conda/activate.d/env_vars.sh
export LD_LIBRARY_PATH=$CONDA_PREFIX/lib/python3.9/site-packages/nvidia/cudnn/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$CONDA_PREFIX/lib:$LD_LIBRARY_PATH
export PATH=$CONDA_PREFIX/bin:$PATH
export XLA_FLAGS=--xla_gpu_cuda_data_dir=$CONDA_PREFIX
```

7. Install Tensorrt and export PATH:

```
pip install nvidia-tensorrt==8.4.1.5
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CONDA_PREFIX/lib/python3.9/site-packages/tensorrt
```

Tensorflow installation on Alpine [NVIDIA] (3)



8. We link libnvinfer.so.8 to libnvinfer.so.7 :

```
ln -s $CONDA_PREFIX/lib/python3.9/site-packages/tensorrt/libnvinfer.so.8 $CONDA_PREFIX/lib/python3.9/site-packages/t
ln -s $CONDA_PREFIX/lib/python3.9/site-packages/tensorrt/libnvinfer_plugin.so.8  $CONDA_PREFIX/lib/python3.9/site-pa
```

9. To test that your installation is working you will need to exit "acompile" first and load on the NVIDIA GPU debug partition on Alpine"

```
conda deactivate
exit
sinteractive --partition=atesting_a100 --qos=testing --time=00:05:00 --gres=gpu:1 --ntasks=2
module load anaconda
conda activate tf_env
python3 -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"
```

10. Make sure to exit the GPU debug node partition after testing the installation.

```
$ exit
exit
```

Containers (1)

- Apptainer/Singularity can now be built directly on the cluster
- Can be built either from a definition file or converted from a docker image.

- e.g: `apptainer build –f R_spack_env.sif R_spack_env.def`

Containers (2)

- module load singularity
- export ALPINE_SCRATCH=/gpfs/alpine1/scratch/$USER
- export SINGULARITY_TMPDIR=$ALPINE_SCRATCH/singularity/tmp
- export SINGULARITY_CACHEDIR=$ALPINE_SCRATCH/singularity/cache
  mkdir -pv $SINGULARITY_CACHEDIR $SINGULARITY_TMPDIR