Assignment 0

Logging In

To access the Greene HPC cluster, you need to be on the NYU network. If you're off-campus, connect via the NYU VPN.

Steps to Log In

1. **Open a Terminal** on your local machine.
2. **Connect via SSH** (replace `[netid]` with your NYU NetID):

```
Local ---> Greene login node ---> Greene compute node (NOT USING FOR THIS COURSE)
                              |
                              ---> Burst node ---> GCP compute node
```

```
ssh [netid]@greene.hpc.nyu.edu
ssh burst
```

Understanding the Filesystem

The Greene HPC cluster has different directories optimized for various storage needs.

Directory	Variable	Purpose	Flushed After	Quota
/archive	$ARCHIVE	Long-term storage	No	2TB / 20k inodes
/home	$HOME	Configurration Files	No	50GB / 30k inodes
/scratch	$SCRATCH	Temporary data storage	Yes(60 days)	5TB / 1M inodes

- **Check Your Quota:**

```
myquota
```

- **Recommended:** Store the data you want to keep in `/scratch/[netid]` and temporary data in `/tmp`.

Running Interactive Jobs

When you need to run scripts or perform debugging interactively, follow the steps below.

Typical Workflow

1. Log in: Greene's login node.
2. Log in to Burst node.
3. Request a job / computational resource and wait until Slurm grants it.

- You always need to request a job for GPUs.

4. Execute singularity and start container instance.
5. Activate conda environment with your own deep learning libraries.
6. Run your code, make changes/debugging.

Accounts and Partitions

- **Account:** `csci_ua_0480_042-2025fa`
- **Partitions:**
  - `interactive` (for lightweight tasks)
  - `n2c48m24`
  - `g2-standard-12`
  - `g2-standard-48`
  - `c12m85-a100-1`
  - `c24m170-a100-2`
  - `n1s8-t4-1`
  - `n1s8-v100-1`
  - `n1s16-v100-2`

Understanding Partitions

- **Partitions** are specific resources or queues on the cluster.

Simple Scripts and File Operations

For non-GPU tasks, use the `interactive` partition.
Requesting an Interactive Session:

```
srun --account=csci_ua_0480_042-2025fa --partition=interactive --time=1:00:00 --pty /bin/bash
```

- **Options:**
  - `--account`: Specify account.
  - `--partition`: Choose partition.
  - `--time`: The time limit for which you want the shell for.
  - `--pty /bin/bash`: Open an interactive shell.

> **Tip:** After allocation, verify the node:

```
hostname
```

GPU Access

For GPU tasks, request resources from a GPU partition. Each student is assigned to Slurm account with 200 GPU hours (12000 minutes) and sufficient CPU time.
Requesting a GPU Session:

```
srun --account= --partition=g2-standard-12 --gres=gpu:1 --time=1:00:00 --pty /bin/bash
```

- **Options:**
  - `--gres=gpu:1`: Request one GPU.
  - `--time=1:00:00`: Set time limit.

  Verify GPU Allocation:

```
nvidia-smi
```

Monitoring Jobs

Check the status of your jobs in the Slurm queue.
Check Your Jobs:

```
squeue -u [netid]
```

Cancel a Job:

- **Exit the Session:** Press `Ctrl+D` or type `exit`.
- Use

  ```
  scancel [jobid]
  ```

Setting Up Singularity and Conda

Copying the Filesystem Image

Copy the empty filesystem image (once per semester).
Get on a GPU Node:

```
srun --account=csci_ua_0480_042-2025fa --partition=g2-standard-12 --gres=gpu:1 --time=04:00:00 --pty /bin/bash
```

Navigate to Scratch Directory:

```
cd /scratch/[netid]
```

Download Overlay Filesystem:

```
scp greene-dtn:/scratch/work/public/overlay-fs-ext3/overlay-25GB-500K.ext3.gz .
```

Filesystems can be mounted as read-write (`rw`) or read-only (`ro`) when we use it with singularity.

- read-write: use this one when setting up env (installing conda, libs, other static files)
- read-only: use this one when running your jobs. It has to be read-only since multiple processes will access the same image. It will crash if any job has already mounted it as read-write.

Unzipping the Image

Unzip the ext3 filesystem (takes about 5 minutes).

```
gunzip -vvv ./overlay-25GB-500K.ext3.gz
```

```
# Copy the appropriate singularity image to the current working directory
scp -rp greene-dtn:/scratch/work/public/singularity/cuda11.8.86-cudnn8.7-devel-ubuntu22.04.2.sif .
```

Installing Conda

Install Conda inside the Singularity container.
Start Singularity:

```
singularity exec --nv --bind /scratch --overlay  /scratch/[netid]/overlay-25GB-500K.ext3:rw /scratch/[netid]/cuda11.8.86-cudnn8.7-devel-ubuntu22.04.2.sif
/bin/bash
```

What the above command means

- `/scratch/[netid]/overlay-25GB-500K.ext3:rw` This is the path to the overlay file that should be used by the singularity container. All libraries that you install will be stored here. In case you have installed the overlay file in a different directory, then please update it accordingly. For example, in case your overlay file is stored in a directory like `/scratch/[netid]/ComputerVision`, then you must update the overlay argument to `/scratch/[netid]/ComputerVision/overlay-25GB-500K.ext3:rw`. Furthermore, please note the flag **:rw** This indicates that the overlay is in read-write mode. This mode should only be used when you are downloading/installing libraries. Otherwise please change the flag to **:ro** (read-only mode).
- Similarly `/scratch/[netid]/cuda11.8.86-cudnn8.7-devel-ubuntu22.04.2.sif` is the path to the .sif file used as the singularity image.

Inside Singularity:
Once you enter the singularity shell, your terminal should look something like this:
`Singularity>`

To check what your path is currently within the Singularity shell, use the **pwd** command

For storing your files, and program data, do ensure that you are working in the **/scratch/[netid]** directory.

Download and install conda

```
cd /ext3/
wget --no-check-certificate https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-Linux-x86_64.sh
sh Miniforge3-Linux-x86_64.sh -b -p /ext3/miniforge3
```

Create wrapper script

```
touch /ext3/env.sh
echo '#!/bin/bash' >> /ext3/env.sh
echo 'unset -f which' >> /ext3/env.sh
echo 'source /ext3/miniforge3/etc/profile.d/conda.sh' >> /ext3/env.sh
echo 'export PATH=/ext3/miniforge3/bin:$PATH'         >> /ext3/env.sh
echo 'export PYTHONPATH=/ext3/miniforge3/bin:$PATH'   >> /ext3/env.sh
```

Activate conda environment

```
source /ext3/env.sh
```

Update Conda and Install Packages

```
conda config --remove channels defaults
conda update -n base conda -y
conda clean --all --yes
```

```
conda create -n computer_vision python==3.9
conda activate computer_vision
conda install pip --yes
conda install ipykernel --yes
conda install pytorch
```

Testing the Setup

Test PyTorch and GPU access:

```
python

>>> import torch
>>> torch.cuda.is_available()
True
>>> x = torch.tensor([1, 2])
>>> x
tensor([1, 2])
```

Running Batch Jobs

For longer experiments or multiple jobs, use batch jobs.

Batch Job Workflow

1. **Log In** to Greene.
2. **Submit an `sbatch` Script**.

Submitting a Job Script

```
Request an interactive shell
```

Write the Batch Script:

```
#!/bin/bash
#SBATCH --job-name=job_wgpu
#SBATCH --account=csci_ua_0480_042-2025fa
#SBATCH --partition=g2-standard-12
#SBATCH --open-mode=append
#SBATCH --output=./%j_%x.out
#SBATCH --error=./%j_%x.err
#SBATCH --export=ALL
#SBATCH --time=1:00:00
#SBATCH --gres=gpu:1

singularity exec --nv --bind /scratch --nv --overlay  /scratch/[netid]/overlay-25GB-500K.ext3:ro /scratch/[netid]/cuda11.8.86-cudnn8.7-devel-
ubuntu22.04.2.sif /bin/bash -c "
source /ext3/env.sh
conda activate computer_vision
cd /scratch/[netid]/computer_vision/
python ./test_script.py
"
```

Submit Batch Job:

```
sbatch gpu_job.slurm
```

Check Job Status:

```
squeue -u [netid]
```

Checking Job Output

After job completion, check the output log.

SCP Tutorial How to copy files around?

- Use Git and you don't need any of these :)
- From local to Greene, on local run

```
scp [optional flags] [file-path] [netid]@greene.hpc.nyu.edu:[greene-destination-path]
```

- From Greene to local, on local run

```
scp [optional flags] [netid]@greene.hpc.nyu.edu:[file-path] [local-destination-path]
```

- From Greene to GCP, on GCP run

```
scp [optional flags] greene-dtn:[file-path] [gcp-destination-path]
```

- From GCP to Greene, on GCP run

```
scp [optional flags] [file-path] greene-dtn:[greene-destination-path]
```

- From local to GCP: local → Greene → GCP
- From GCP to local: GCP → Greene → Local

Running Jupyter Notebook

Create jupyter kernel

```
mkdir -p ~/.local/share/jupyter/kernels
cd ~/.local/share/jupyter/kernels
scp -r greene-dtn:/share/apps/mypy/src/kernel_template ./my_env
cd ./my_env

ls
#kernel.json  logo-32x32.png  logo-64x64.png  python
```

In the 'python' file, change the singularity command at the bottom to

```
singularity exec $nv \
  --overlay /scratch/[netid]/overlay-25GB-500K.ext3:ro \
  /scratch/[netid]/cuda11.8.86-cudnn8.7-devel-ubuntu22.04.2.sif \
  /bin/bash -c "source /ext3/env.sh; conda activate computer_vision; $cmd $args"
```

Edit the default kernel.json file by setting PYTHON_LOCATION and KERNEL_DISPLAY_NAME.

```
{
 "argv": [
  "/home/[netid]/.local/share/jupyter/kernels/my_env/python", #PYTHON_LOCATION
  "-m",
  "ipykernel_launcher",
  "-f",
  "{connection_file}"
 ],
 "display_name": "my_env", #KERNEL_DISPLAY_NAME
 "language": "python"
}
```

Using GCP post setup

1. ssh into Greene:
    1. `ssh [netid]@greene.hpc.nyu.edu`
2. ssh into a burst shell:
    1. `ssh burst`
3. request a compute node:
    1. `srun --account=csci_ua_0480_042-2025fa --partition=g2-standard-12 --gres=1 --time=04:00:00 --pty /bin/bash`
4. Enter your singularity shell:
    1. `singularity exec --nv --bind /scratch --nv --overlay /scratch/[netid]/overlay-25GB-500K.ext3:rw /scratch/[netid]/cuda11.8.86-cudnn8.7-devel-ubuntu22.04.2.sif /bin/bash`
5. Activate the conda environment
    1. `source /ext3/env.sh`
    2. `conda activate computer_vision`
6. Go to your working directory (in `/scratch/[netid]/`)
    1. `cd /scratch/[netid]`
    2. `cd ComputerVision/Assignment_x`
7. Run your python files:
    1. `python test.py`

Note

- In case you do use a GPU compute node, and do provide the `--nv` flag in your singularity command and are unable to detect pytorch, please check which version of pytorch (cpu or gpu) is installed on your overlay and accordingly update your packages.
- Please be careful while copy pasting commands from this file.

Go to https://ood-burst-001.hpc.nyu.edu/ to run Jupyter Notebook and VS Code.
Troubleshooting: https://sites.google.com/nyu.edu/nyu-hpc/training-support/general-hpc-topics/tunneling-and-x11-forwarding
Here are some additional references (Note: while these are for Greene HPC, the same should work within your cloudburst shells as well):

1. https://sites.google.com/nyu.edu/nyu-hpc/training-support/general-hpc-topics/slurm-submitting-jobs?authuser=0
2. https://sites.google.com/nyu.edu/nyu-hpc/hpc-systems/greene/software/open-ondemand-ood-with-condasingularity?authuser=0

Acknowledgement: Thanks to Divyam Madan for providing the base version for this tutorial.