NYU

HPC Basics

Advanced Topics in Embodied Learning and Vision

Ying Wang

2025.01.23

SSH

How to install vpn:
https://www.nyu.edu/life/information-technology/infrastructure/network-services/vpn.html

1. If you are in NYU network or VPN

   $ ssh <net-id>@greene.hpc.nyu.edu

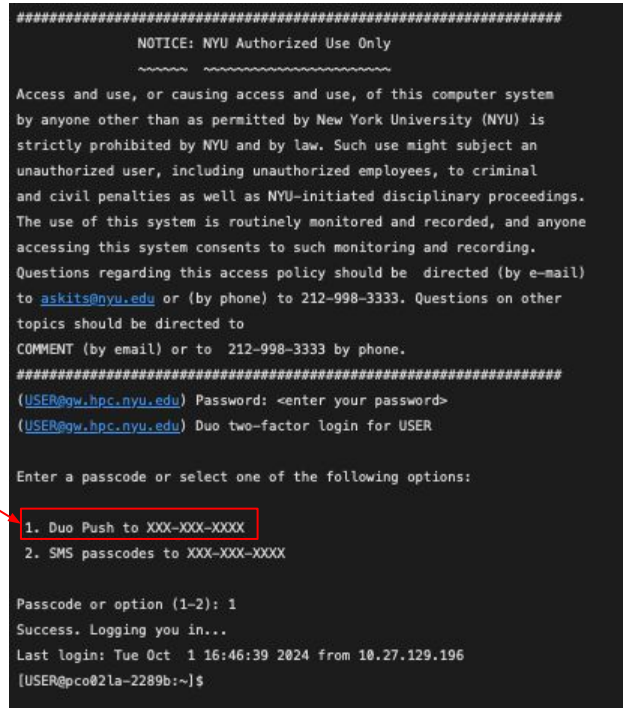
2. If you are outside NYU,

ssh to gateway (require Duo MFA)

   $ ssh <net-id>@gw.hpc.nyu.edu

from gateway, ssh to greene

   $ ssh <net-id>@greene.hpc.nyu.edu



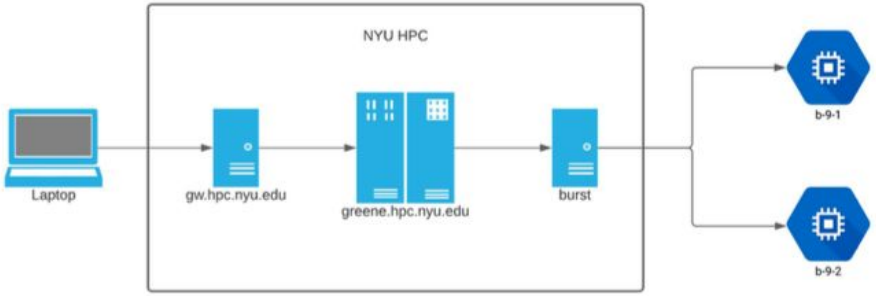
SSH Setup

(base) yingw@b-10-27-94-204 ~ % vim .ssh/config

```
Host nyugateway
 User yw3076
 Hostname gw.hpc.nyu.edu
 ForwardAgent yes
 ControlPath ~/.ssh/.%r@%h:%p
 ControlMaster auto
 ControlPersist yes

Host greene
 User yw3076
 Hostname greene.hpc.nyu.edu
 ForwardAgent yes
 StrictHostKeyChecking no
 IdentityFile ~/.ssh/id_rsa
 UserKnownHostsFile /dev/null

Host greeneburst
 User yw3076
 Hostname log-burst.hpc.nyu.edu
 ForwardAgent yes
 ProxyJump greene
```



ssh-keygen (generate keys if you haven't done so already)

~/.ssh/id_rsa.pub (stores your public key)

• Copy your public key to the remote server for authentication using "ssh-copy-id greene"

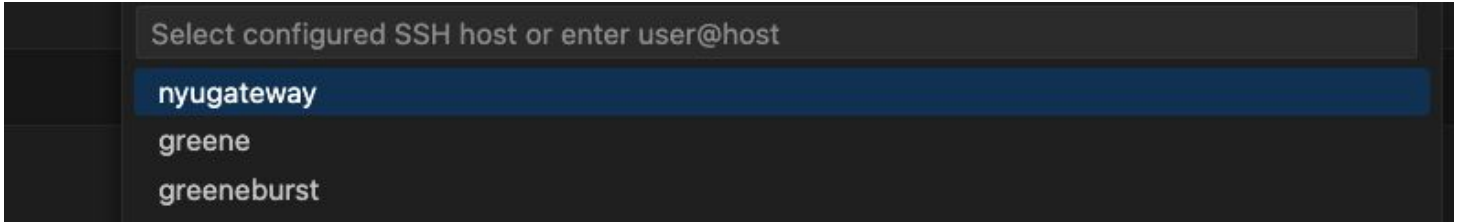
• Once the server receives your public key and considers it trustworthy, the server marks the key as authorized in authorized_keys

~/.ssh/id_rsa (stores your private key)

• The possession of this private key is proof of user's identity. Store it carefully.

VSCode

In VS Code, select Remote-SSH: Connect to Host...



https://code.visualstudio.com/docs/remote/ssh

Access Cloud Bursting

Each student has with 200 GPU hours and sufficient CPU time.

1. From a Greene login node

   ssh burst

2. CPU-only interactive job

   srun --account=csci_ga_3033-2025sp --partition=interactive --time=04:00:00 --pty /bin/bash

   1 V100 GPU

srun --account=csci_ga_3033-2025sp --partition=n1s8-v100-1 --gres=gpu:v100:1 --time=04:00:00 --pty /bin/bash

```
CSCI_GA_3033_2025sp = {
  accounts = { "csci_ga_3033-2025sp" },
  partitions = { "interactive", "n2c48m24",
                 "n1s8-v100-1",
                 "g2-standard-12",
                 "c12m85-a100-1",
                 "n1s8-t4-1",
                 "g2-standard-48",
                 "n1s16-v100-2",
                  "c24m170-a100-2"}
```

Env Setup (Singuarlity & Overlay)

- Copy the empty fs gzip to your scratch path (e.g. /scratch/<NETID>/ or $SCRATCH for your root scratch):

  cp /scratch/work/public/overlay-fs-ext3/overlay-50G-10M.ext3.gz $SCRATCH/

- Unzip the archive:

  gunzip -v $SCRATCH/overlay-50G-10M.ext3.gz (can take a while to unzip...)

- Execute container with overlayfs (check comment below about rw arg):

  singularity exec --overlay $SCRATCH/overlay-50G-10M.ext3:rw
  /scratch/work/public/singularity/cuda10.1-cudnn7-devel-ubuntu18.04-20201207.sif /bin/bash

  -rw (read-write): use this one when setting up env
  -ro (read-only): use this one when running your jobs.

- Check file systems: df -h. There will be a record: overlay 53G 52M 50G 1% /. The size equals to the filesystem image you chose. The actual content of the image is mounted in /ext3.
- (optional) Create a file in overlayfs: touch /ext3/testfile
- Exit from Singularity

Env Setup (Conda)

- Start a CPU (GPU if you want/need) job:

  srun --nodes=1 --tasks-per-node=1 --cpus-per-task=1 --mem=32GB --time=1:00:00 --gres=gpu:1 --pty /bin/bash

- Start singularity (notice --nv for GPU propagation):

  singularity exec --nv --overlay $SCRATCH/overlay-50G-10M.ext3:rw
  /scratch/work/public/singularity/cuda10.1-cudnn7-devel-ubuntu18.04-20201207.sif /bin/bash

- Install conda:

  wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh

  bash ./Miniconda3-latest-Linux-x86_64.sh -b -p /ext3/miniconda3

- Install your conda env in /ext3:

  conda create -n tutorial python=3.10

  conda activate tutorial

- Install packages

  conda install pytorch torchvision cudatoolkit=10.2 -c pytorch

  conda install jupyterlab

- Exit singularity container (not the CPU/GPU job)

Submit a Job

Interactive job: srun --mem=64G --time=1:00:00 --gres=gpu:a100:1 --pty /bin/bash

Submit a sbatch job: sbatch xxx.slurm

Create a file named *xxx.slurm*. Inside the file:

```
#!/bin/bash
#SBATCH --job-name=finetune
#SBATCH --open-mode=append
#SBATCH --output=/scratch/yw3076/llm_clustering/outputs/%x_hf_%j.out
#SBATCH --error=/scratch/yw3076/llm_clustering/outputs/%x_hf_%j.err
#SBATCH --export=ALL
#SBATCH --time=24:00:00
#SBATCH --mem=128G
#SBATCH --mail-type=ALL
#SBATCH --mail-user=yw3076@nyu.edu
#SBATCH --gres=gpu:1
#SBATCH -c 4

singularity exec --nv --overlay /scratch/yw3076/overlay-50G-10M.ext3:ro  /scratch/work/public/singularity/cuda12.2.2-cudnn8.9.4-devel-ubuntu22.04.3.sif /bin/bash -c "
source /ext3/env.sh
conda activate llava
bash /scratch/yw3076/llm_clustering/lmms-finetune/scripts/llava_ft_$1.sh $2 $3
"
```

If you are using burst,

- don't forget to specify account and partition. [check slide 5 to see available partitions]
- Always add #SBATCH --requeue so your jobs will be automatically requeued (Note that the instances might be shut down by Google and we don't have visibility). You need to save checkpoints and enable resuming from checkpoints in your code. [pytorch]

Slurm

squeue -u ${USER} - shows state of your jobs in the queue.

scancel <jobid> - cancel job with specified id. You can only cancel your own jobs.

scancel -u ${USER} - cancel all your current jobs, use this one very carefully.

scontrol hold <jobid> - hold pending job from being scheduled. This may be helpful if you noticed that some data/code/files are not ready yet for the particular job.

scontrol release <jobid> - release the job from hold.

scontrol requeue <jobid> - cancel and submit the job again.

Greene Storage

Very small! Be careful!

```
1. mkdir $SCRATCH/python_cache
2. cd
3. ln -s $SCRATCH/python_cache/.cache
```

Space	Environment Variable	Space Purpose	Backed Up / Flushed	Quota Disk Space / # of Files
/home	$HOME	Personal user home space that is best for small files	YES / NO	50 GB / 30 K
/scratch	$SCRATCH	Best for large files	NO / Files not accessed for 60 days	5 TB / 1 M
/archive	$ARCHIVE	Long-term storage	YES / NO	2 TB / 20 K
HPC Research Project Space	NA	Shared disk space for research projects	YES / NO	Payment based TB-year/inodes-year
/vast	$VAST	Flash memory for high I/O workflows	NO / Files not accessed for 60 days	2 TB / 5 M

Move your project to /archive after completion

Check your usage:

$myquota

```
[yw3076@log-1 yw3076]$ myquota

Hostname: log-1 at Thu Jan 16 06:36:01 PM EST 2025

Filesystem   Environment   Backed up?   Allocation       Current Usage
Space        Variable      /Flushed?    Space / Files    Space(%) / Files(%)

/home        $HOME         Yes/No       50.0GB/30.0K       0.93GB(1.85%)/6731(22.44%)
/scratch     $SCRATCH      No/Yes        5.0TB/1.0M     2701.97GB(52.77%)/392706(39.27%)
/archive     $ARCHIVE      Yes/No        2.0TB/20.0K      248.18GB(12.12%)/7643(38.22%)
/vast        $VAST      NO/YES        2TB/5.0M           0.0TB(0.0%)/293231(5%)
```

Greene Public Datasets

The HPC team makes available a number of public sets that are commonly used in analysis jobs.

- /scratch/work/public/ml-datasets/
- /vast/work/public/ml-datasets/

We recommend to use version stored at /vast (when available) to have better read performance

Many datasets are available in the form of '.sqf' file, which can be used with Singularity.

```
$ singularity exec \
  --overlay /<path>/pytorch1.8.0-cuda11.1.ext3:ro \
  --overlay /vast/work/public/ml-datasets/coco/coco-2014.sqf:ro \
  --overlay /vast/work/public/ml-datasets/coco/coco-2015.sqf:ro \
  --overlay /vast/work/public/ml-datasets/coco/coco-2017.sqf:ro \
  /scratch/work/public/singularity/cuda11.1-cudnn8-devel-ubuntu18.04.sif /bin/bash
```

Some datasets (such as Ego4D and ImageNet) require accepting the terms of license agreement. Complete the forms and send them to HPC team (hpc@nyu.edu) for permission.

Burst File Systems

- Note that Burst and Greene have independent file systems! If you use burst, you cannot access the public dataset on Greene. We also have limited storage on burst and thus not recommend using burst if your project requires large datasets.

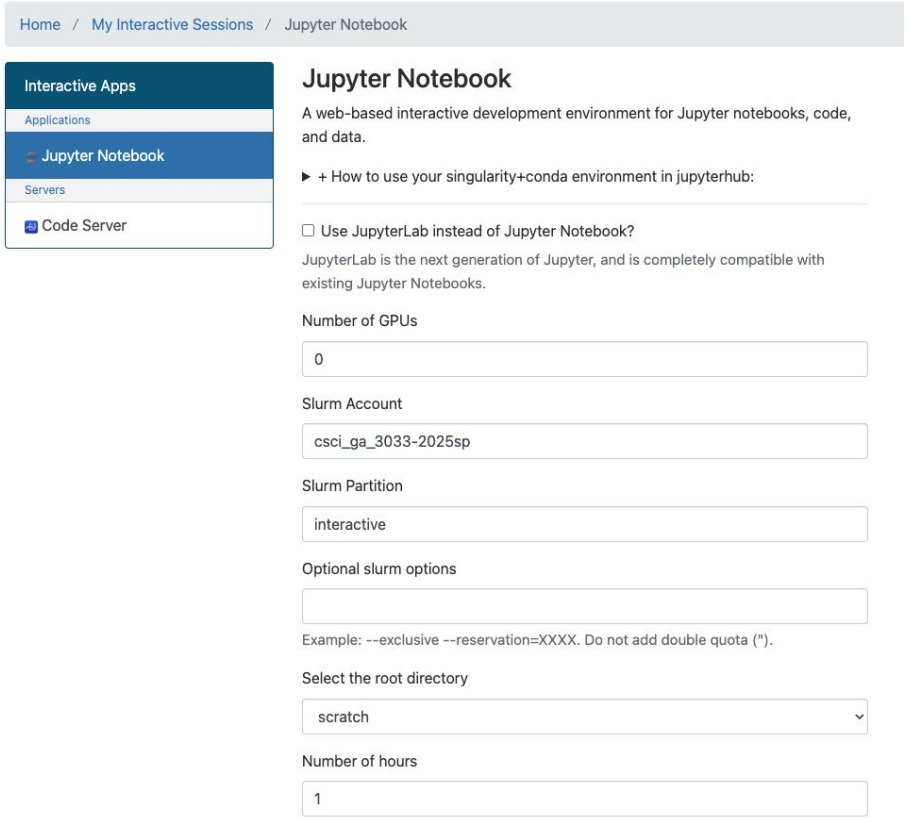
- **Data transfer (Greene –> Burst):** Greene Data transfer nodes is available with hostname greene-dtn. On a Cloud instances, run scp

  scp -rp greene-dtn:/scratch/work/public/singularity/ubuntu-20.04.3.sif .

- **Data transfer (Local –> Burst):** Transfer the data to Greene first and then repeat the step above.

OOD (Burst)

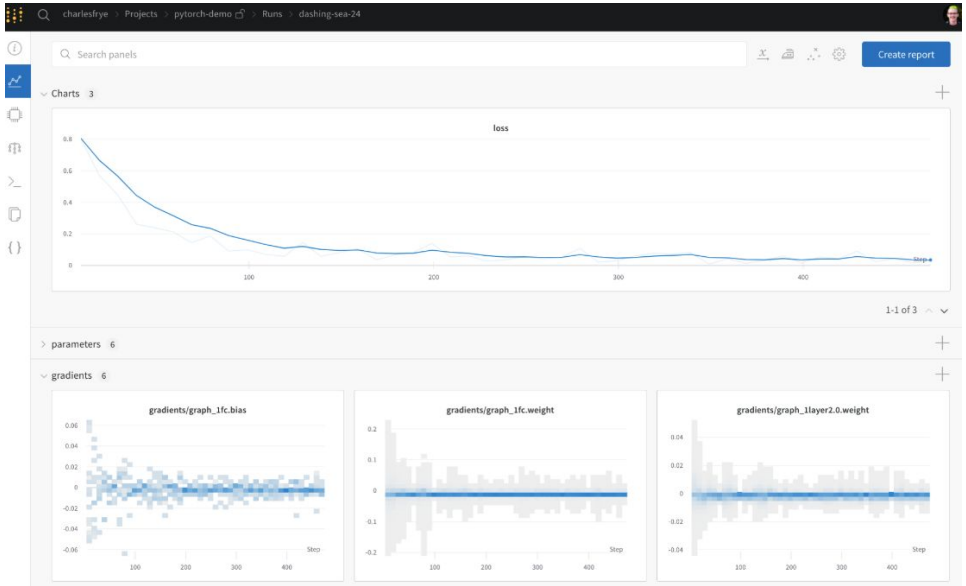
- You might want to use jupyter lab for exploratory work.
- We recommend using jupyter lab via OOD https://ood-burst-001.hpc.nyu.edu/
- Don’t forget to specify account and partition!



Resources

- Greene website
  https://sites.google.com/nyu.edu/nyu-hpc/hpc-systems/greene
- System status (VPN)
  https://sites.google.com/nyu.edu/nyu-hpc/hpc-systems/greene/system-status?authuser=0
- Greene tutorial
  https://github.com/nyu-dl/cluster-support/tree/master/greene
- Google Cloud Bursting
  https://sites.google.com/nyu.edu/nyu-hpc/hpc-systems/cloud-computing/hpc-bursting-to-cloud

Weights and Biases (Optional)

- A useful platform to track your training job on HPC
- Tutorials: https://docs.wandb.ai/tutorials/
- 

In pseudocode, what we'll do is:

```
# import the library
import wandb

# start a new experiment
wandb.init(project="new-sota-model")

# capture a dictionary of hyperparameters with config
wandb.config = {"learning_rate": 0.001, "epochs": 100, "batch_size": 128}

# set up model and data
model, dataloader = get_model(), get_data()

# optional: track gradients
wandb.watch(model)

for batch in dataloader:
  metrics = model.training_step()
  # log metrics inside your training loop to visualize model performance
  wandb.log(metrics)

# optional: save model at the end
model.to_onnx()
wandb.save("model.onnx")
```