# Using Greene

All lab assignments will be **executed, tested, and graded** in Greene. This documentation shows you how to use Greene. The commands provided below can be used in Mac Terminal, Windows command line, and Linux terminal.
**Note (for Windows users):** If your Windows laptop does not support SSH in the command line, then you can follow this documentation for setting up SSH in the command line. Alternatively, you can also use PuTTY or WSL2.
To check if ssh is installed:
Follow SSH to Greene, if you get an error message: **ssh is not recognized as an internal or external command, operable program, or batch file.** Then you need to install SSH following the above guides.
**Note:** Whenever prompted for a password, enter your NYU password (the password used for accessing NYU Albert and other services).

# How to use Greene to execute programs

This is the typical flow of using Greene to run your programs. Go through each section below for more detailed instructions.

1. SSH to Greene
2. Running a parallel job
    a. Method-1 is easier to use compared to Method-2. Only use Method-2 if it takes too long to run a parallel job in Method-1.
    b. MPI programs run in the CPU partition, and CUDA programs run in the GPU partition.
3. Transfer your code to Greene.
4. Execute your program. (Compiling and executing code instructions will be provided during the lab assignments.)

If you encounter any difficulties, the "Commonly Faced Issues" section may help you. If that doesn't resolve the problem, you can also seek start discussion on NYU Brightspace.

# SSH to Greene

1. SSH to Greene Jump host

```
ssh netID@gw.hpc.nyu.edu
```

For example, if netID is ab1234

```
ssh ab1234@gw.hpc.nyu.edu
```

2. SSH to Greene cluster

```
ssh netID@greene.hpc.nyu.edu
```

3. SSH to Google Cloud Burst

```
ssh burst
```

**Note:** When SSH to Greene cluster for the first time, you might get this prompt. This key is not known by any other names. Are you sure you want to continue connecting?

   a. **Solution:** Type yes.

# Running a job using Method-1 (for CPU-intensive MPI programs)

To run your program, you will need a coding environment with all the dependencies. It might take some time to create the parallel job (1–5 minutes).

1. Run the parallel job.

   ```
   srun --account=csci_ua_0480_051-2024fa --partition=n2c48m24 --nodes=1
   --tasks-per-node=24 --cpus-per-task=1 --mem=0  --time=04:00:00 --pty
   /bin/bash
   ```

   After this, the prompt will be something like: **bash-5.1$**

2. Run the command below.

   ```
   /share/apps/images/ubuntu-22.04.sif
   ```

   After this, the prompt will look like: **Singularity>**

Now you can execute your program (read transfer files to the Greene section to transfer your program to this machine).

# Running a job using Method-2 (for CPU-intensive MPI programs)

This is only recommended in case you were **unable to run the job using Method-1** or it's **taking too long to get access to using Method-1**, then you can run the program using this method. In this mode, your program will be in queue, and once the program is executed, you can view the logs.

1. Create job file
   Create a file called file.txt and add the following contents to it. For most of your lab assignments, the content of file.txt will largely stay the same, except for the last line. After /bin/bash -c "...", you'll need to insert the specific compile and run command provided in your lab assignment. Be sure to use the correct file name when doing so.

```
#!/bin/bash

#SBATCH --job-name=program
#SBATCH --output=/home/rr4549/%j_%x.out
#SBATCH --error=/home/rr4549/%j_%x.err
#SBATCH --time=01:00:00
#SBATCH --account=csci_ua_0480_051-2024fa
#SBATCH --partition=n2c48m24
#SBATCH --nodes=1
#SBATCH --tasks-per-node=24
#SBATCH --cpus-per-task=1
#SBATCH --mem=0


singularity exec /share/apps/images/ubuntu-22.04.sif /bin/bash -c
"mpicc -o program program.c; mpirun -np 8 ./program"
```

2. Run the parallel job.

```
sbatch file.txt
```

3. View the status of the job

```
squeue -u netID
```

4. View the result of the job.

   Once the job is not in the queue anymore, you can view the "jobID_jobName.out" and "jobID_jobName.err" file to check the final output of your program. If there are no errors when executing the batch file, then the output logs will be in "jobID_jobName.out"; otherwise, the error logs will be in the "jobID_jobName.err" file.
   To view these files, you can view them at “Files” section in https://ood-burst-001.hpc.nyu.edu/. Ensure that you are connected to NYU VPN (Mac, Windows) or connected to NYU wifi. (Screenshots for reference)

# Transfer files to Greene

To transfer your program files to Greene. If there are multiple files, then it is recommended to zip those files and then transfer them.

1. Using GUI **(recommended)**
   a. Connect to NYU VPN (Mac, Windows) or to the NYU wifi.
   b. Login to https://ood-burst-001.hpc.nyu.edu/ and go to Files -> Home Directory. Here you can upload and download files, and it will directly be available for you in Greene. (Screenshots for reference)
   c. Now you have the file in the machine, go to your terminal and can execute your code.

2. Using SCP
   If you are unable to install NYU VPN or just interested in transferring via terminal, then follow the steps below:
   a. Copy from local machine to Greene - Execute the command below on your machine

```
scp -o ProxyJump=netID@gw.hpc.nyu.edu /path/to/file/in/local_machine
netID@greene.hpc.nyu.edu:/home/netID/
```

   For example:

```
scp -o ProxyJump=rr4549@gw.hpc.nyu.edu
/Desktop/RahulRaman_rr4549_lab1.zip
rr4549@greene.hpc.nyu.edu:/home/rr4549/
```

   b. Copy from Greene to Google Cloud Burst
   SSH to Greene, run the parallel job and execute the command below

```
scp -rp greene-dtn:/home/netID/file .
```

   Now you have the file in the machine and can execute your code.

# Running a job in Method-1 (for GPU-intensive CUDA programs)

To run your program, you will need a coding environment with all the dependencies. It might take some time to create the parallel job (1–5 minutes).

3. Run the parallel job.

```
srun --account=csci_ua_0480_051-2024fa --partition=n1s8-v100-1
--gres=gpu:v100:1 --time=01:00:00 --pty /bin/bash
```

   After this, the prompt will be something like: **bash-5.1$**

4. Run the command below.

```
/share/apps/images/run-cuda-12.2.2.bash
```

   After this, the prompt will look like: **Singularity>**

Now you can execute your program (read transfer files to the Greene section to transfer your program to this machine).

# Running a job in Method-2 (for GPU-intensive CUDA programs)

This is only recommended in case you were **unable to run the job using Method-1** or it's **taking too long to get access using Method-1**, then you can run the program using this method. In this mode, your program will be in queue, and once the program is executed, you can view the logs.

1. Create job file
   Create a file called file.txt and add the following contents to it. For most of your lab assignments, the content of file.txt will largely stay the same, except for the last line. After /bin/bash -c "...", you'll need to insert the specific compile and run command provided in your lab assignment. Be sure to use the correct file name when doing so.

```
#!/bin/bash

#SBATCH --job-name=program
#SBATCH --output=/home/rr4549/%j_%x.out
#SBATCH --error=/home/rr4549/%j_%x.err
#SBATCH --time=01:00:00
#SBATCH --account=csci_ua_0480_051-2024fa
#SBATCH --partition=n1s8-v100-1
#SBATCH --gres=gpu:v100:1


singularity exec /share/apps/images/run-cuda-12.2.2.bash /bin/bash -c
"nvcc hello_world.cu -o hello_world_batch;./hello_world_batch"
```

2. Run the parallel job.

```
sbatch file.txt
```

3. View the status of the job

```
squeue -u netID
```

4. View the result of the job.

   Once the job is not in the queue anymore, you can view the "jobID_jobName.out" and "jobID_jobName.err" file to check the final output of your program. If there are no errors when executing the batch file, then the output logs will be in "jobID_jobName.out"; otherwise, the error logs will be in the "jobID_jobName.err" file.
   To view these files, you can view them at “Files” section in https://ood-burst-001.hpc.nyu.edu/. Ensure that you are connected to NYU VPN (Mac, Windows) or connected to NYU wifi. (Screenshots for reference)

# Commonly faced Issues

In case you face any issues, please use the Brightspace discussion. Here are some common issues and their solutions:

2. **Issue:** When SSH to Greene cluster for the first time, you might get this prompt. This key is not known by any other names. Are you sure you want to continue connecting?
    a. **Solution:** Type yes.

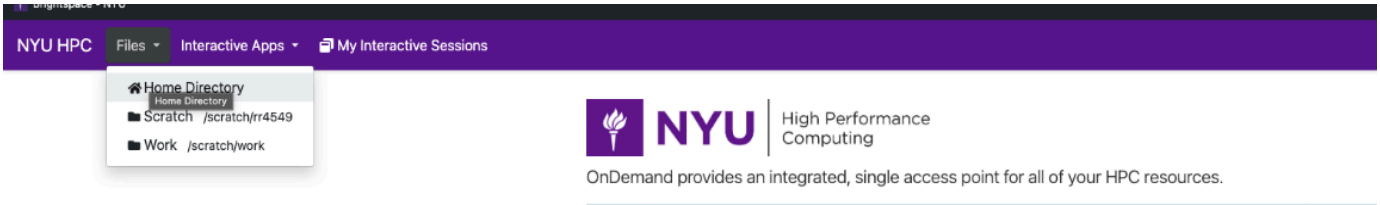
3. **Issue:** After some time of inactivity, you might get a broken pipe error.
    a. **Solution:** ssh to Greene cluster, then ssh burst, and then run the parallel job again.

4. **Issue:** Sometimes even when you enter the correct password to enter the Greene Cluster, it still says incorrect password.
    a. **Solution:** Close the terminal and start over SSH again.

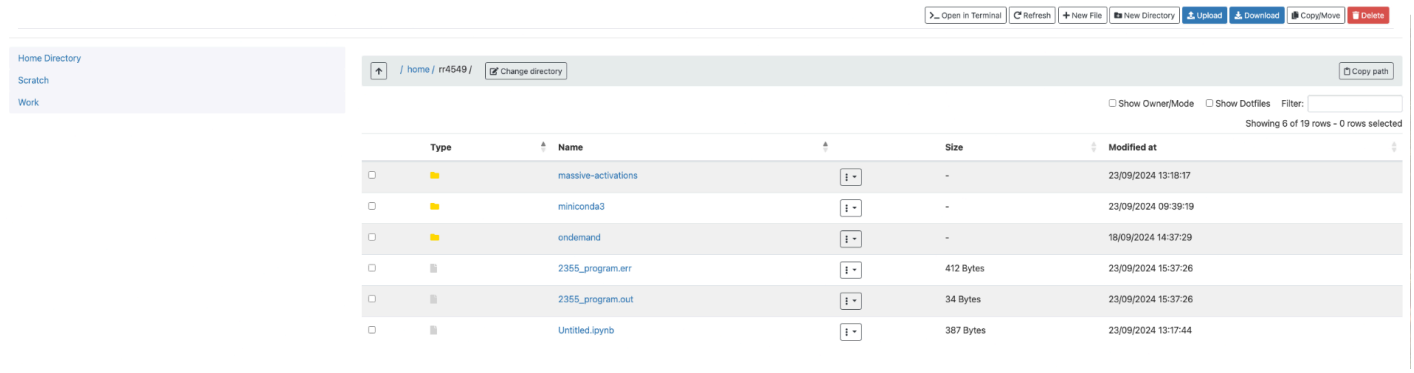
5. **Issue:** Sometimes you might get this WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!
    a. **Solution:** run the commands below on your computer.
        i. ssh-keygen -R gw.hpc.nyu.edu
        ii. ssh-keygen -R greene.hpc.nyu.edu

# Using cloud burst OOD GUI

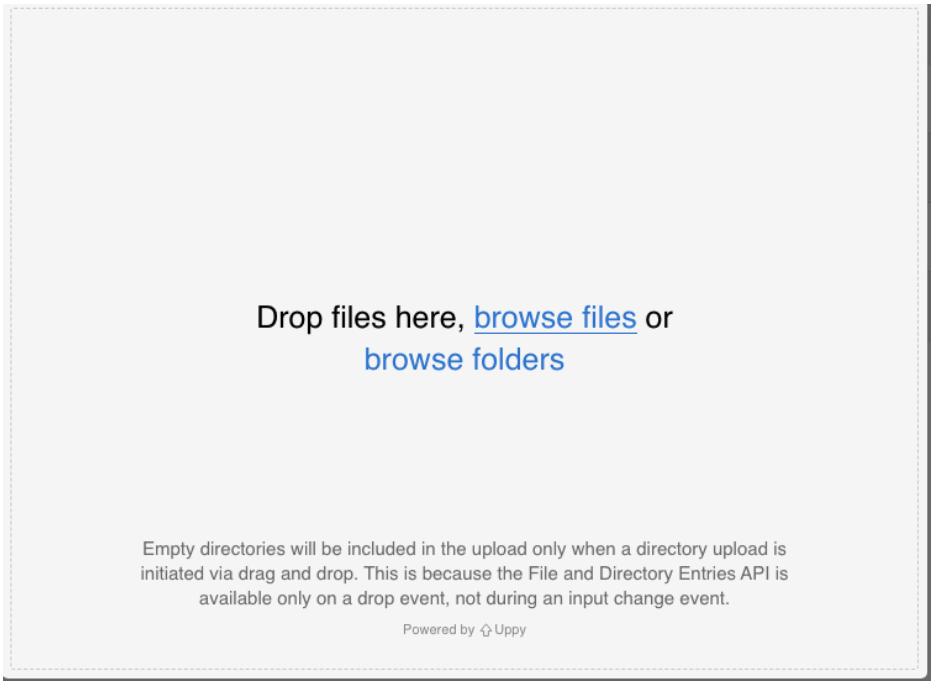
Select Files, go to Home Directory.



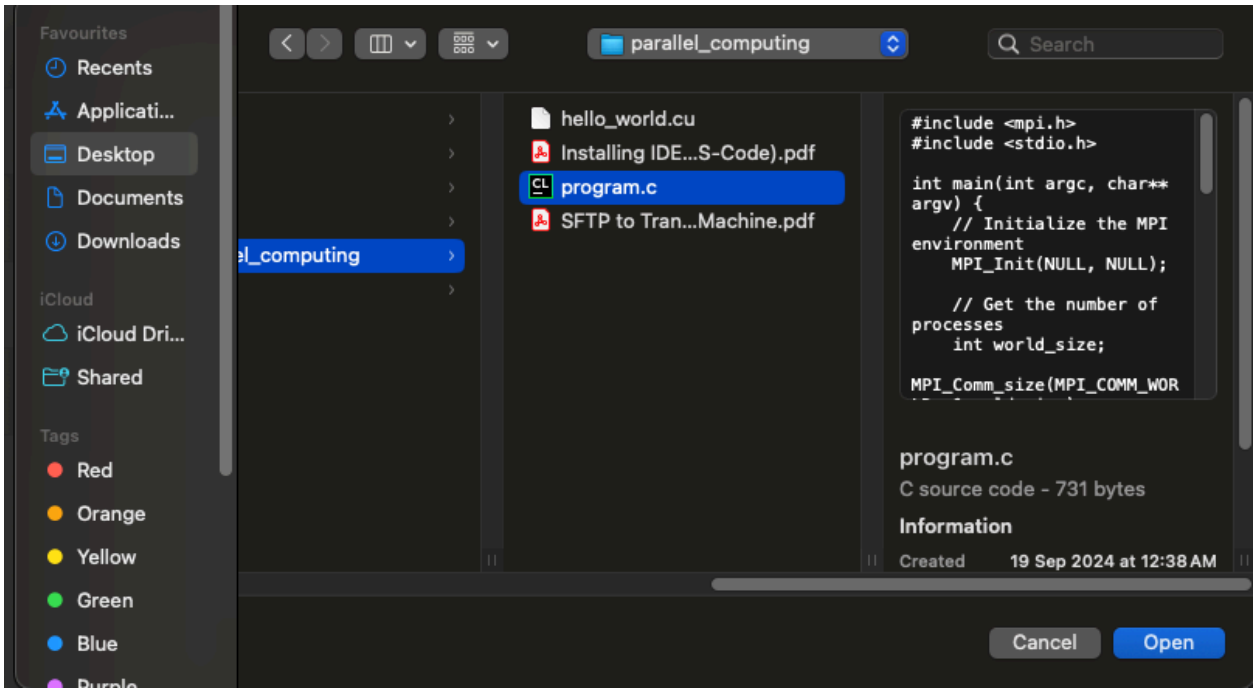

You can upload files to the machine by clicking on the upload button.



Once you click on the upload button, this pop-up window will appear. Click on browse files.




Select your file to upload to Greene.



Then click on Upload 1 file.


You can also view your files on your machine. You can also view the contents of your file by clicking on the file. For example to view the output of my Method-2 job with job-id 2355, just click on 2355_program.out