These instructions are also available at:

Accessing NERSC’s Systems

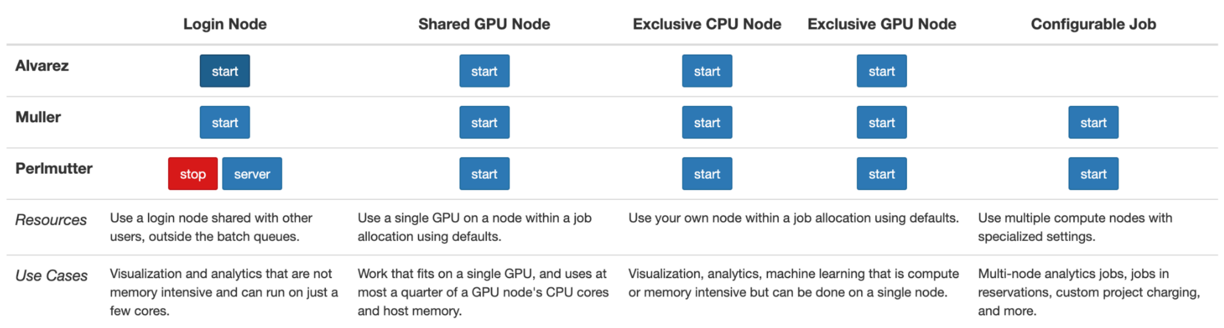
You have two options: Jupyter or SSH:

1. (jupyter) Go to: https://jupyter.nersc.gov/ and sign in with your credentials from step 2
2. (ssh) In a Unix/Linux terminal, type:
   ```
   elvis@laptop> ssh < user>@perlmutter.nersc.gov
   ```

Accessing the Terminal from within Jupyter

You might not want to install a terminal emulator and ssh – in that case, you can access a terminal window on jupyter.nersc.gov (step 3 above) as follows:

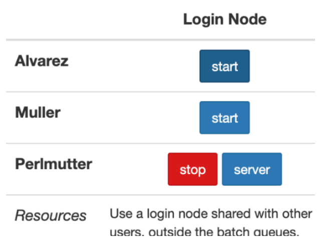
3. After logging into jupyter.nersc.gov (step 3) you should see something like this:



	Login Node	Shared GPU Node	Exclusive CPU Node	Exclusive GPU Node	Configurable Job
Alvarez	start	start	start	start	
Muller	start	start	start	start	start
Perlmutter	stop server	start	start	start	start
Resources	Use a login node shared with other users, outside the batch queues.	Use a single GPU on a node within a job allocation using defaults.	Use your own node within a job allocation using defaults.		Use multiple compute nodes with specialized settings.
Use Cases	Visualization and analytics that are not memory intensive and can run on just a few cores.	Work that fits on a single GPU, and uses at most a quarter of a GPU node's CPU cores and host memory.	Visualization, analytics, machine learning that is compute or memory intensive but can be done on a single node.		Multi-node analytics jobs, jobs in reservations, custom project charging, and more.

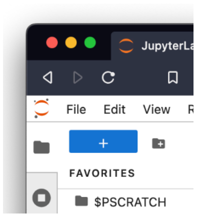
(You might not see the bright red “stop” button, and probably fewer rows/columns – that’s OK)

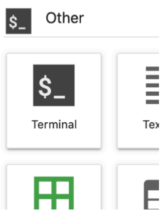
4. Select “Server” in the “Login Node” column and “Perlmutter” row



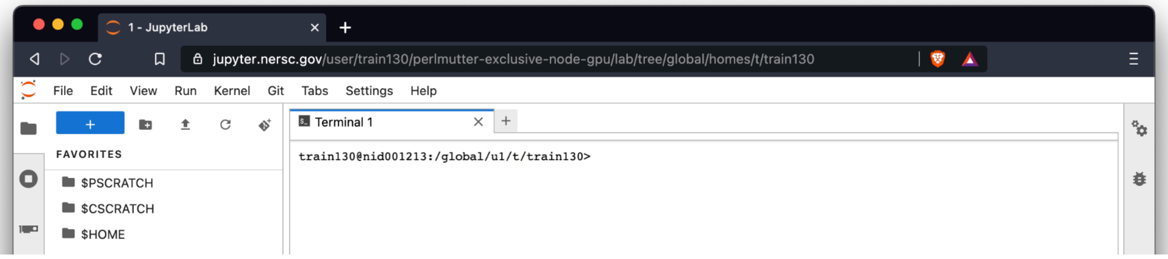
	Login Node
Alvarez	start
Muller	start
Perlmutter	stop server
Resources	Use a login node shared with other users, outside the batch queues.

5. After a short while, you should see a blue button (with a “+” sign) in the top left hand corner. Push it, and then select “Terminal” (you might need to scroll)





6. If you did everything correctly, you should see a terminal window in the left-hand tab:


1 - JupyterLab
jupyter.nersc.gov/user/train130/perlmutter-exclusive-node-gpu/lab/tree/global/homes/t/train130
File Edit View Run Kernel Git Tabs Settings Help
Terminal 1
FAVORITES
$PSCRATCH
$CSCRATCH
$HOME

```
train130@nid001213:/global/u1/t/train130>
```

Installing the Jupyter Kernels at NERSC

This tutorial requires specialized kernels to be installed – that's an automated process which you need to initiate in your training account

7. Access a shell on Perlmutter (either step 4, or step 5-8)
8. Clone the tutorial repository:
   ```
   elvis@login12> git clone
   git@github.com:JuliaParallel/julia-hpc-tutorial-lanl25.git
   ```
9. Enter the tutorial folder:
   ```
   elvis@login12> cd julia-hpc-tutorial-lanl25
   ```
10. Run the install script:
    ```
    elvis@login12> ./install.sh
    ```
    (this might take some time – that's OK)

Running Gray-Scott on Perlmutter/NERSC

These instructions are for the Perlmutter system at NERSC and are based on the ORNL Gray Scott Tutorial (qr code after step 18, below)

Configuring Gray-Scott on Perlmutter

13. Access a shell on Perlmutter (either step 4, or step 5-8)
14. Obtain Gray-Scott from GitHub, first access your scratch area and create a user-specific directory, clone the repository pointing at the GraySccott-JACC branch.

```
14.1 cd $SCRATCH
14.2 mkdir $USER
14.3 cd $USER
14.4 git clone --branch GrayScott-JACC
https://github.com/JuliaORNL/GrayScott.jl.git
```

15. Run the script prepared for this tutorial
    GrayScott.jl/scripts/config_perlmutter.sh
    to set up modules, environment, and packages.
    ```
    source GrayScott.jl/scripts/config_perlmutter.sh
    ```

Running Gray-Scott jobs on Perlmutter

16. Create an area for Gray-Scott runs outside the repository (e.g. run001, future runs will be in run002, run003, etc.)
    ```
    mkdir run001
    ```

17. Copy the Gray-Scott settings file and the job_perlmutter.sh to the run directory
    ```
    17.1 cp GrayScott.jl/examples/settings-files.json run001
    17.2 cp GrayScott.jl/scripts/job_perlmutter.sh run001
    ```

18. Submit your first job to Perlmutter. It should generate an adios bp file output, and total runtime should be around 12 seconds using a single MPI process and NVIDIA GPU.
    ```
    18.1 cd run001
    18.2 sbatch job_perlmutter.sh
    ```

To validate the output, please refer to the ORNL Gray Scott Tutorial

Slurm Jobscript Cheatsheet

These are the parameters you need to give to Slurm in order to submit jobs as a training account user.

Note: the GPU node reservation goes from 1:30pm - 5pm ET, Sun Nov 17th. During the reservation you should use:

19. To use 1 GPU only (sample flags for `sbatch` or `salloc`) use Slurm parameters
    ```
    -A ntrain1 --reservation=lanl_training -C gpu -N 1 -c 32 -G 1 -t
    30:00 -q shared
    ```

20. To use multiple nodes (sample flags for `sbatch` or `salloc`) use Slurm parameters:

```
-A ntrain1 --reservation=lanl_training -C gpu -N 2 -t 30:00 -q
regular
```

Outside of the reservation, you should use:

21. To use 1 GPU only (sample flags for `sbatch` or `salloc`) use Slurm parameters
    ```
    -A ntrain1 -C gpu -N 1 -c 32 -G 1 -t 30:00 -q shared
    ```

22. To use multiple nodes (sample flags for `sbatch` or `salloc`) use Slurm parameters:
    ```
    -A ntrain1 -C gpu -N 2 -t 30:00 -q regular
    ```