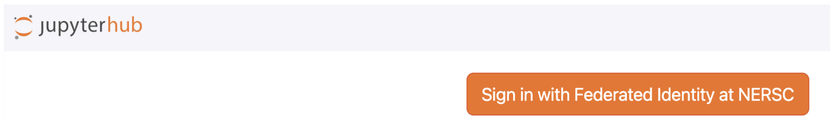
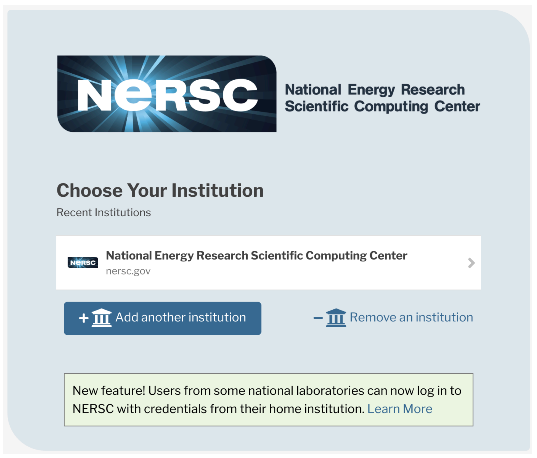
Instructions for Using Perlmutter v1:

Logging In:

1. Go to jupyter.nersc.gov
2. Login using Federated Identity and click National Energy Research Scientific Computing Center:




NERSC
National Energy Research Scientific Computing Center

Choose Your Institution

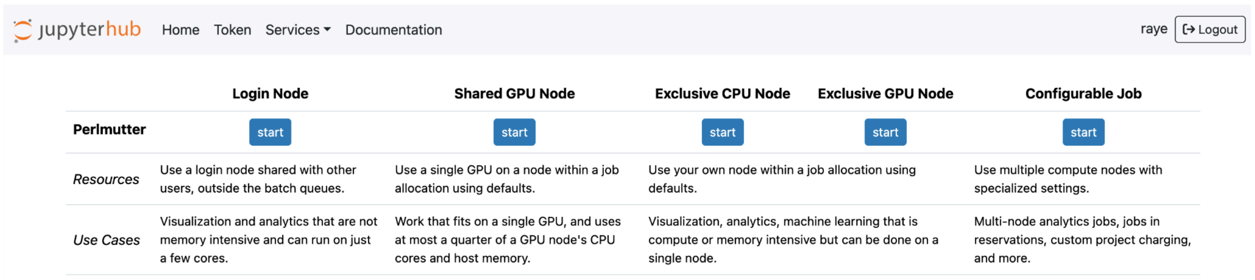
Recent Institutions

National Energy Research Scientific Computing Center
nersc.gov

+ Add another institution
– Remove an institution

New feature! Users from some national laboratories can now log in to NERSC with credentials from their home institution. Learn More

3. You should arrive at a screen that looks like, which will launch JupyterLab on different node types on Perlmutter:



	Login Node	Shared GPU Node	Exclusive CPU Node	Exclusive GPU Node	Configurable Job
Perlmutter	start	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.

   a. **Login Node:** A login node is where you may submit batch scripts or compile code. These nodes are ***shared*** and therefore you must be absolutely sure you are not running compute heavy or memory intensive code on login nodes. ***You should not need login nodes until we are in the distributed computing unit!***
   b. **Shared GPU Node:** A shared GPU node allocates one of 4 GPUs (A100s) that are available on each GPU node. ***If you start a shared GPU node you should ensure that you are using a Jupyter Kernel which has <= 16 threads since you are sharing the 64-core CPU with up to 3 other users.***

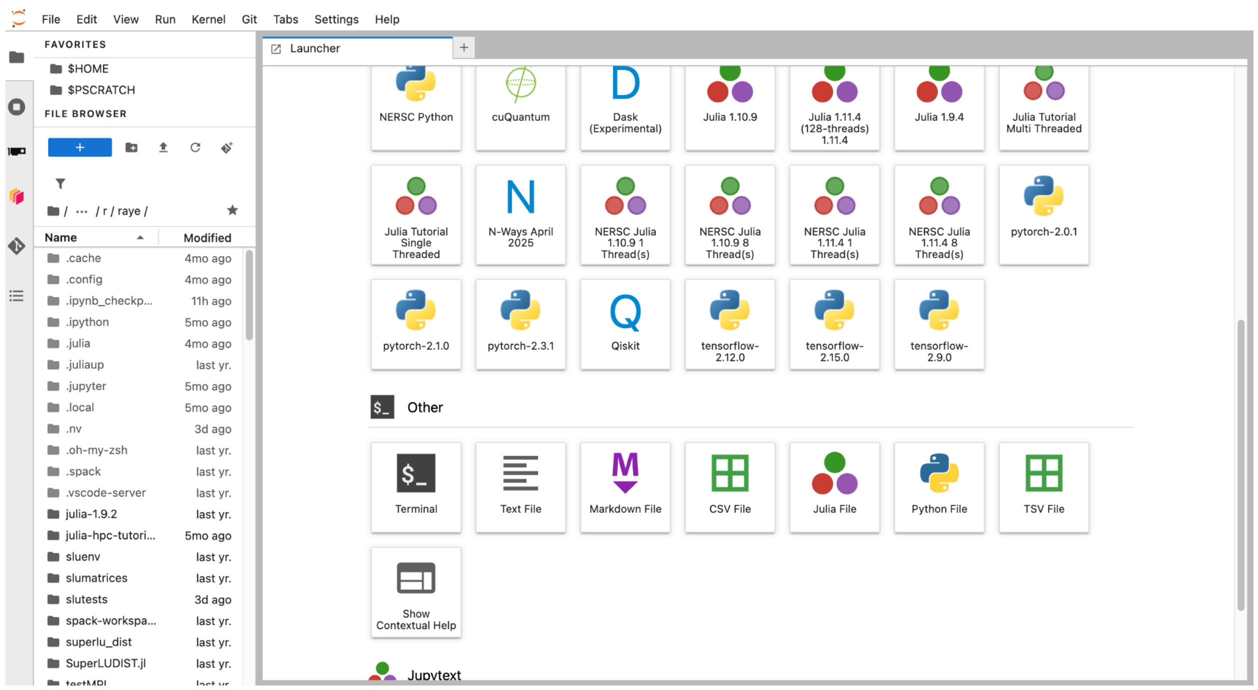
c. **Exclusive CPU Node**: This allocates a CPU node to you which has 128 cores across two CPUs and ~512GB of memory.
d. **Exclusive GPU Node**: This allocates a full GPU node to you which has 64 cores on one CPU, 256GB of RAM and 4 A100 GPUs. ***For homework 2 it is likely best to use a shared GPU Node so you are not wasting multiple GPUs***.
e. **Configurable Jobs:** This will be used later in the course, but it allows you to allocate multiple nodes at once, or more precise configurations.

JupyterLab and Kernels

Required Setup

This is required setup DO NOT SKIP

1. Start a Login Node JupyterLab session, you should see a window that looks like:



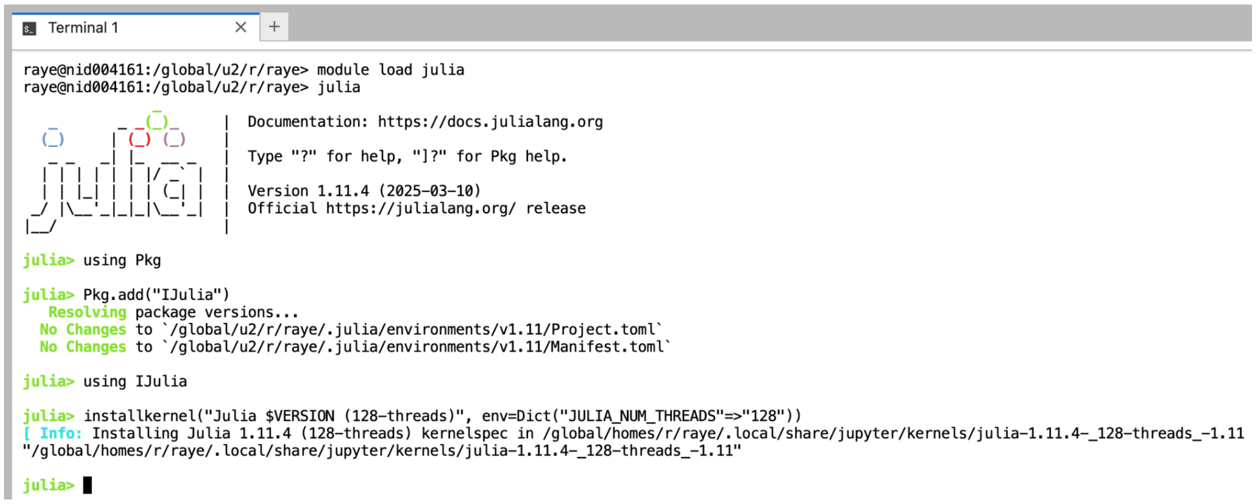
Name	Modified
.cache	4mo ago
.config	4mo ago
.ipynb_checkp...	11h ago
.ipython	5mo ago
.julia	4mo ago
.juliaup	last yr.
.jupyter	5mo ago
.local	5mo ago
.nv	3d ago
.oh-my-zsh	last yr.
.spack	last yr.
.vscode-server	last yr.
julia-1.9.2	last yr.
julia-hpc-tutori...	5mo ago
sluenv	last yr.
slumatrices	last yr.
slutests	3d ago
spack-workspa...	last yr.
superlu_dist	last yr.
SuperLUDIST.jl	last yr.

   a. You may open multiple tabs using the [+] icon. This will open the same launcher screen you see above, and you can use it to open a terminal session, a jupyter notebook, and more.

2. Open a new Terminal window and run the following two commands:

```
module load julia
julia
```

3. Next within Julia you will run the following code starting with `using Pkg`:



```
raye@nid004161:/global/u2/r/raye> module load julia
raye@nid004161:/global/u2/r/raye> julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.11.4 (2025-03-10)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia> using Pkg

julia> Pkg.add("IJulia")
   Resolving package versions...
  No Changes to `/global/u2/r/raye/.julia/environments/v1.11/Project.toml`
  No Changes to `/global/u2/r/raye/.julia/environments/v1.11/Manifest.toml`

julia> using IJulia

julia> installkernel("Julia $VERSION (128-threads)", env=Dict("JULIA_NUM_THREADS"=>"128"))
[ Info: Installing Julia 1.11.4 (128-threads) kernelspec in /global/homes/r/raye/.local/share/jupyter/kernels/julia-1.11.4-_128-threads_-1.11
"/global/homes/r/raye/.local/share/jupyter/kernels/julia-1.11.4-_128-threads_-1.11"

julia> █
```

This will create a new Jupyter kernel with 128 threads. **You must ensure that you only use this kernel on exclusive nodes, otherwise use NERSC default kernels up to 8-threads**.

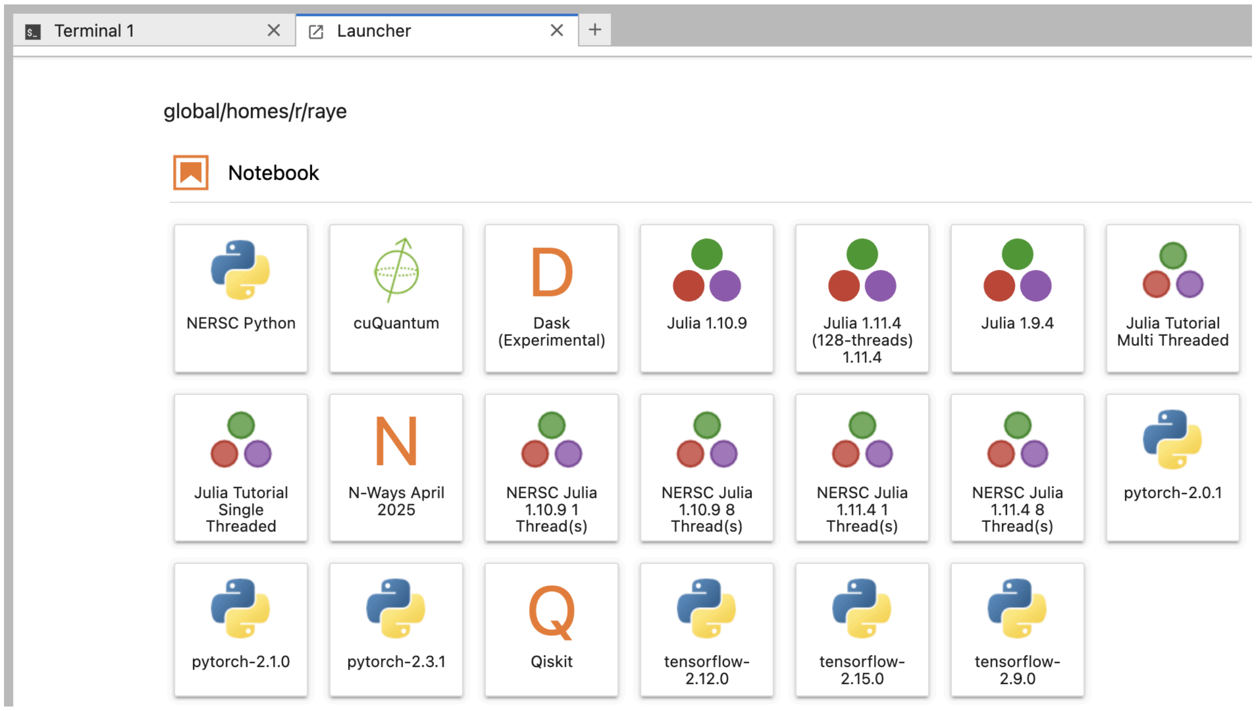
4. Run the same `installkernel` command, but call it `Julia-Exclusive-GPU` and set the `JULIA_NUM_THREADS` variable to 64 (since our ***exclusive*** GPU nodes only have 64 threads)
5. Once you have done this, close this jupyterlab session by going to File > Hub Control Panel and clicking the red **stop** button.

Running Notebooks and Best Practices

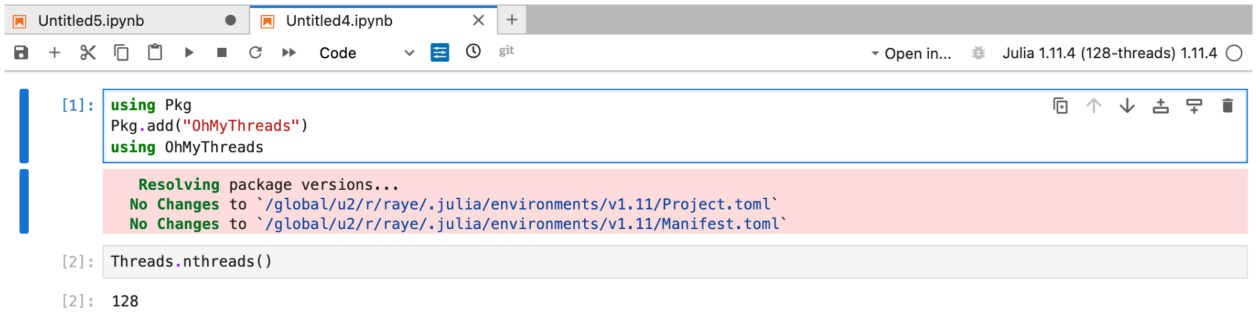
Now when you wish to start a new HPC session simply return to the Hub Control Panel (jupyter.nersc.gov) and start a new allocation. For example if we are running heavy multithreaded code we will follow this workflow:

Exclusive-CPU node notebook

1. Start an exclusive CPU node. Note that you may have to wait or restart your request at different times if the machine is under heavy load.
2. In the launcher you have two options:
    a. First is to launch a Terminal session and start julia from the REPL using `julia --threads=auto.`
    b. However for homework we typically prefer notebooks, so you should click the correct Julia kernel. On Exclusive CPU nodes this will be the kernel you set up above: **Julia 1.11.4 (128-threads)**. On exclusive GPU nodes you should run a 64-thread kernel, and on shared nodes use the default 1 or 8 threads kernels provided by NERSC.



3. Once you are in your notebook you should ensure you are using OhMyThreads with (note that if you always use the default Julia environment you only need to run the first 2 lines the first time you use Julia on the cluster):



```
[1]: using Pkg
     Pkg.add("OhMyThreads")
     using OhMyThreads

       Resolving package versions...
      No Changes to `/global/u2/r/raye/.julia/environments/v1.11/Project.toml`
      No Changes to `/global/u2/r/raye/.julia/environments/v1.11/Manifest.toml`

[2]: Threads.nthreads()

[2]: 128
```

4. It is recommended to check the number of threads you have available in each notebook.

Best Practices

1. Never run heavy computation on login nodes. You may be reprimanded if you do.
2. Ensure you close your JupyterLab servers once you are done. Return to the Control Hub and click STOP on your server.
3. Use the correct number of threads! If you start Julia from a Terminal tab you can set `julia --threads=<N>` where N is either a number or `auto`. ***DO NOT START WITH AUTO ON LOGIN NODES OR SHARED NODES.*** If you are using a Jupyter notebook be sure you start with the correct kernel.
4. Email kimmerer@mit.edu for assistance!