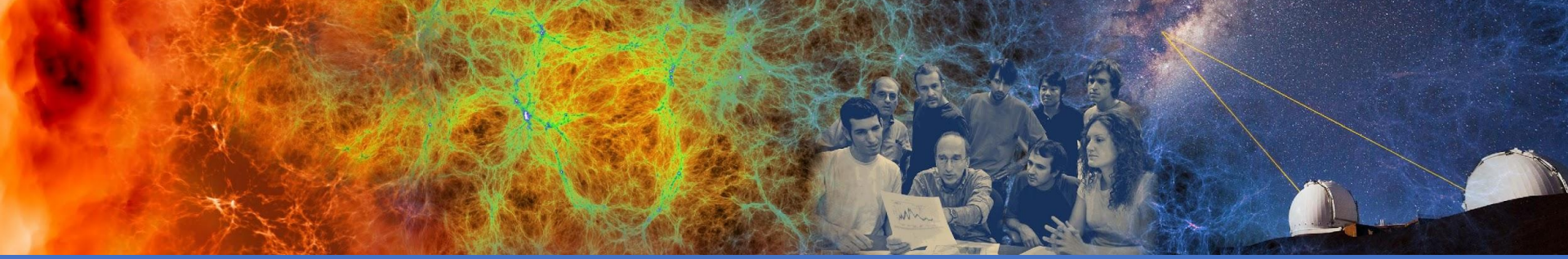
Using Perlmutter and Jupyter for Intro to HPC Bootcamp 2025

Introduction to HPC Bootcamp
Aug 11-15, 2025

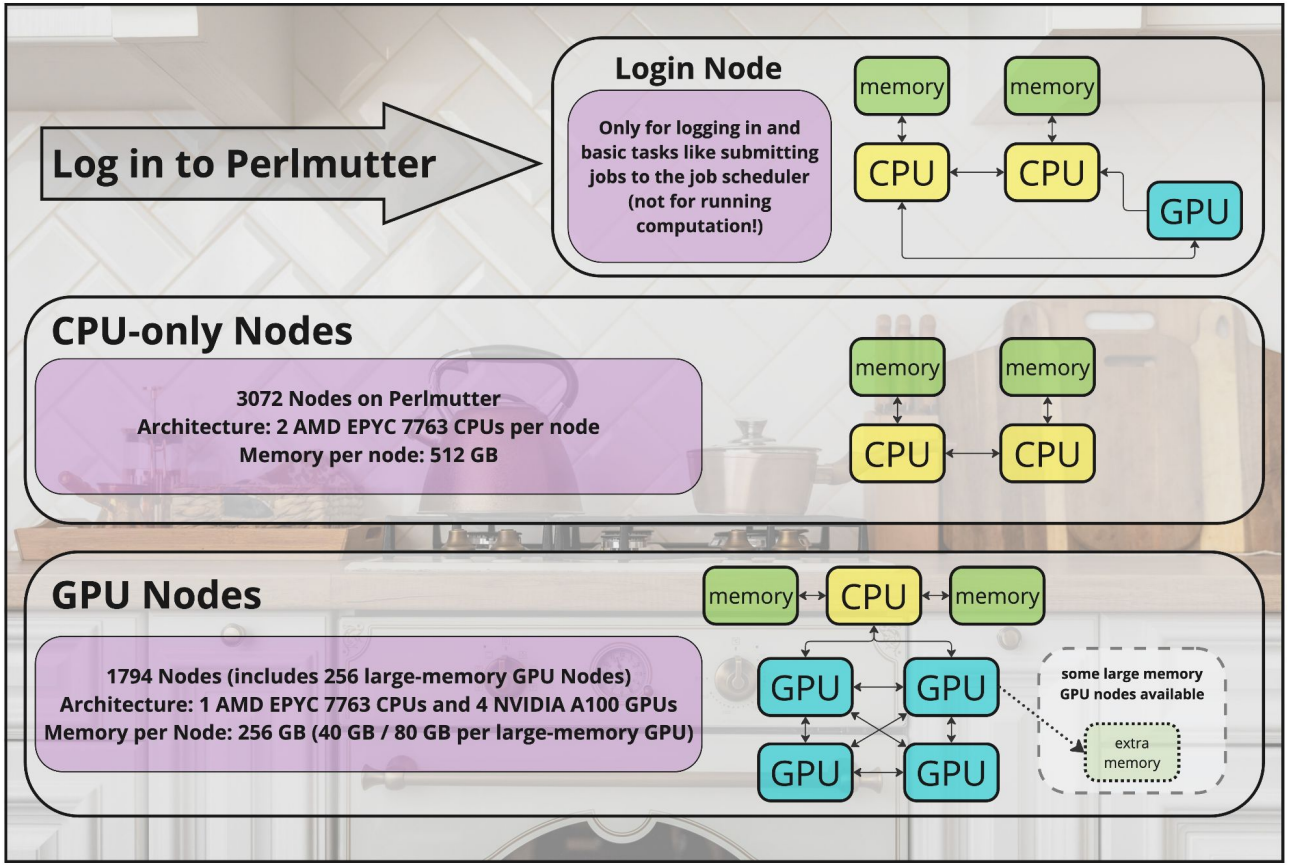
Kelly Rowland, Helen He
NERSC, July 30, 2025

Outline

- Perlmutter Introduction
- Using Jupyter
- File systems
- Compile and run jobs



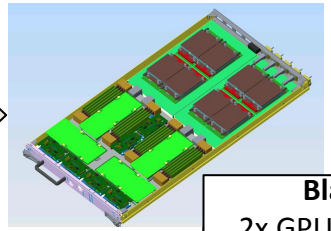
Perlmutter Introduction

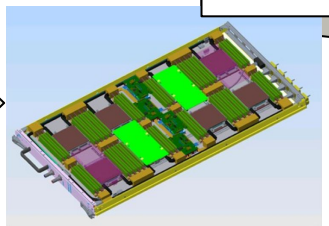

Log in to Perlmutter
Login Node
Only for logging in and basic tasks like submitting jobs to the job scheduler (not for running computation!)
memory
memory
CPU
CPU
GPU
CPU-only Nodes
3072 Nodes on Perlmutter
Architecture: 2 AMD EPYC 7763 CPUs per node
Memory per node: 512 GB
memory
memory
CPU
CPU
GPU Nodes
1794 Nodes (includes 256 large-memory GPU Nodes)
Architecture: 1 AMD EPYC 7763 CPUs and 4 NVIDIA A100 GPUs
Memory per Node: 256 GB (40 GB / 80 GB per large-memory GPU)
memory
CPU
memory
GPU
GPU
GPU
GPU
some large memory GPU nodes available
extra memory

Perlmutter System Configuration

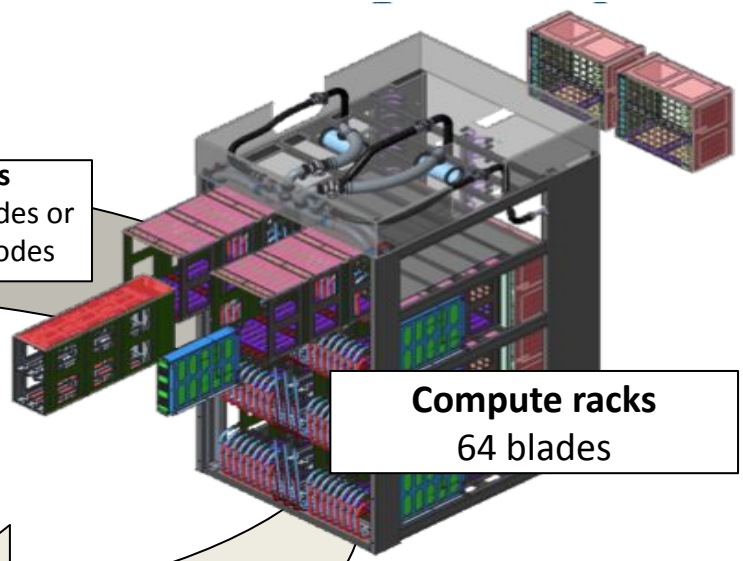
1792 NVIDIA "Ampere" GPU Nodes
4x GPU + 1x CPU (>75 TF)
160 GiB HBM + DDR
4x 200G "Slingshot" NICs

3072 AMD "Milan" CPU Node
2x CPUs
> 256 GiB DDR4
1x 200G "Slingshot" NIC





Blades
2x GPU nodes or
4x CPU nodes



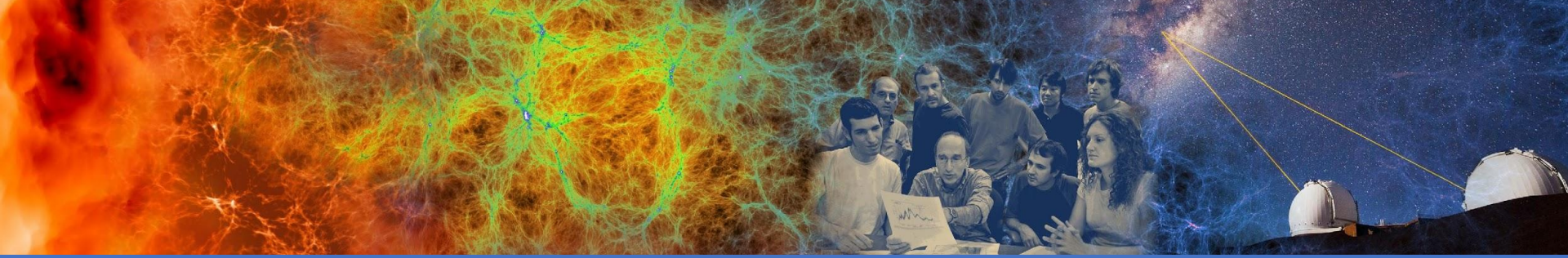
Compute racks
64 blades

Perlmutter in Top 500 list:
- #25 in June 2025
- #5 in Nov 2021

NERSC
BERKELEY LAB
Bringing Science Solutions to the World
U.S. DEPARTMENT OF ENERGY
Office of Science

Connecting to Perlmutter

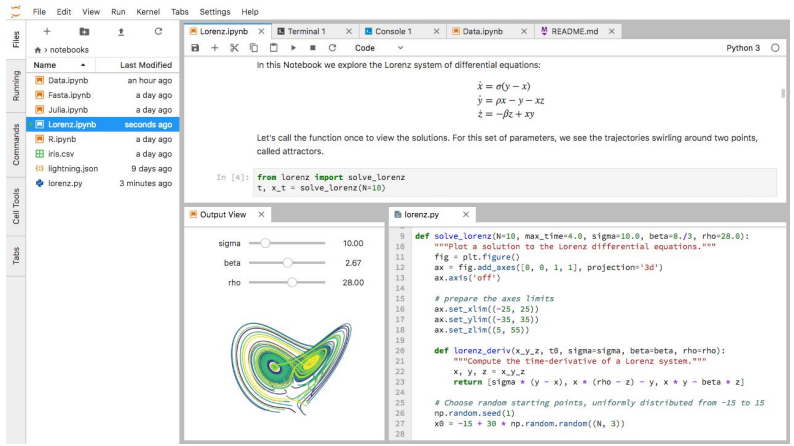
- Set up one-time passwords (OTP) for MFA
- https://docs.nersc.gov/connect/mfa/
- Login to Perlmutter with Jupyter (needs password and OTP)
  - https://www.youtube.com/watch?v=RH8XYGjaEiQ
- Login to Perlmutter with SSH (needs password+OTP)
  - https://www.youtube.com/watch?v=WslPoIlq-oU
- (optional) Set up sshproxy: allows SSH key valid for 24-hr
  - https://docs.nersc.gov/connect/mfa/#sshproxy



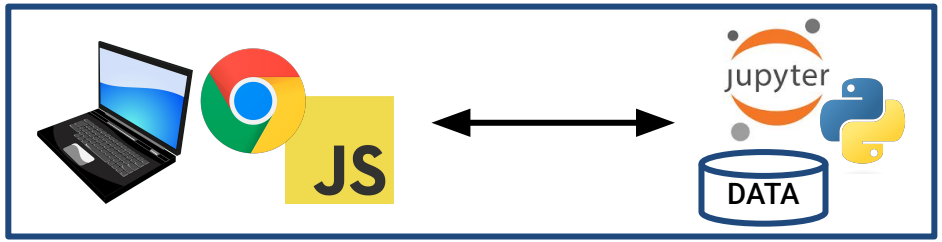
Jupyter at NERSC

What is Jupyter?

- At NERSC, we say “Jupyter” in reference to a collection of many things
  - Access shareable Jupyter “notebooks” via JupyterHub
- What can I put in a Jupyter notebook?
  - Live code
  - Equations
  - Visualizations
  - Narrative text
  - Interactive widgets
- What applications would I use a notebook for?
  - Data cleaning and data transformation
  - Numerical simulation
  - Statistical modeling
  - Data visualization
  - Machine learning
  - Workflows and analytics frameworks



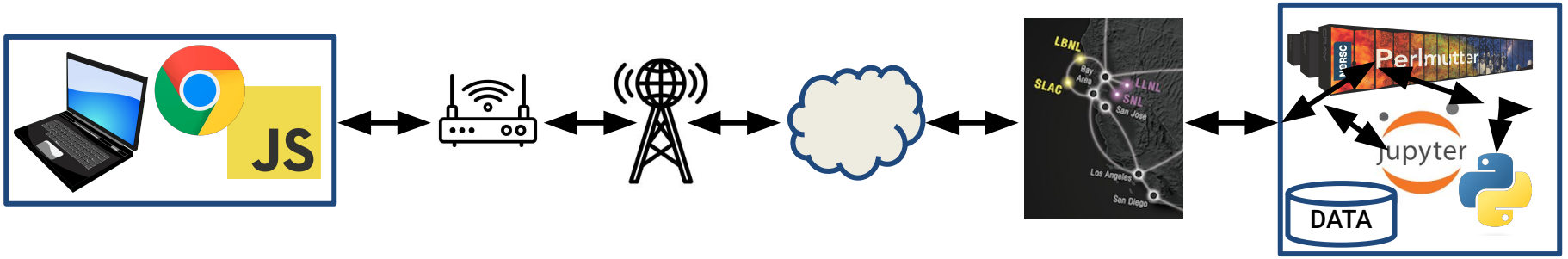
Laptop Jupyter vs HPC Jupyter



A little compute
A little data
All in one place

Laptop

Lots of compute
Lots of data
But way over here

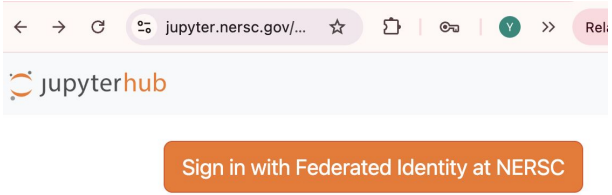


Laptop | Router | ISP | Internet | ESnet | NERSC

Accessing Jupyter at NERSC

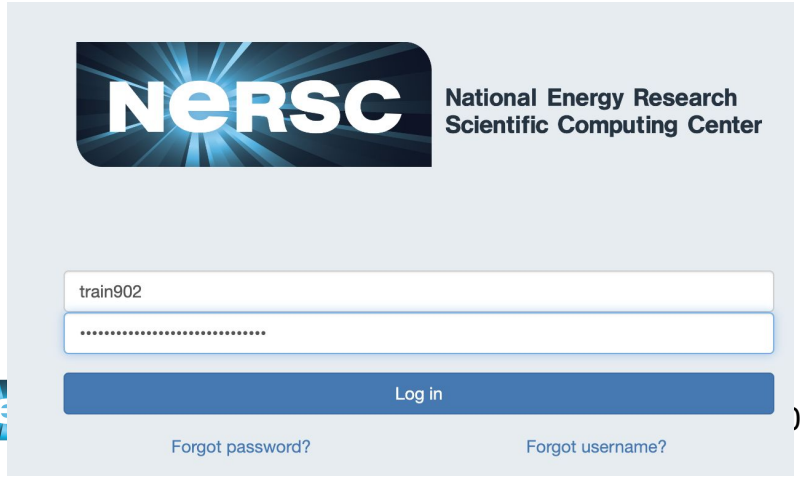
Go to https://jupyter.nersc.gov in any web browser

1)

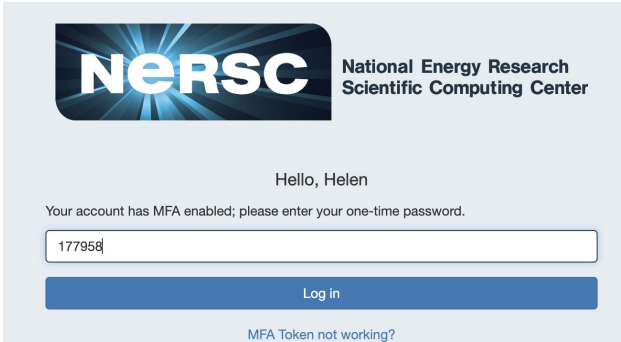


2)

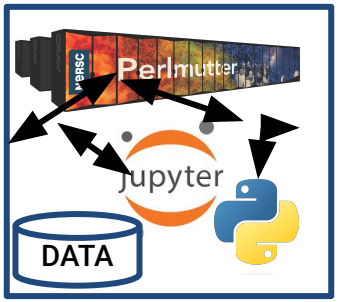
3)



4)



Laptop Jupyter vs HPC Jupyter (II)



NERSC

Laptop: Home directory is “right there” on your laptop
Perlmutter: Home directory is served over (an incredible) network
Many filesystems are served to Perlmutter over network

Great, but, … there’s no free lunch:

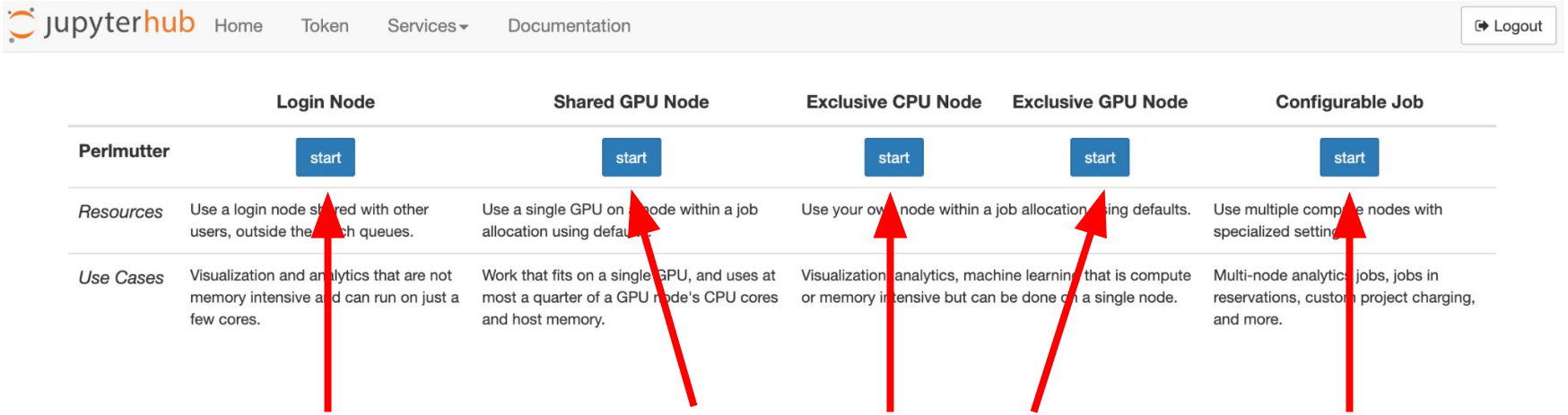
- File system has to look consistent across all those nodes
- I/O has to be coordinated from app to node to network to disk & back
- We use Cray’s “Data Virtualization Service,” and it’s being tuned

Sometimes another user’s file system usage pattern in a running job grabs ahold of DVS and won’t let go!

For Jupyter on compute nodes: Things can slow down, look sluggish, or you may get “gateway timeout” messages. Don’t panic, it’ll recover

Hub Home Page or “Console”

All these options get you running Jupyter on Perlmutter but give you different ways to use its resources.



Use Jupyter on one of Perlmutter’s 40 login nodes

Immediate start, no charging, but more limited resources

Single-click launch of Jupyter on one of Perlmutter’s 4500+ compute nodes

Scheduled, charged, and time-limited, but with GPUs and/or a whole node just to yourself

Customized launch of Jupyter on Perlmutter compute nodes

Scheduled, charged, time-limited, but you control settings directly

Node Reservations

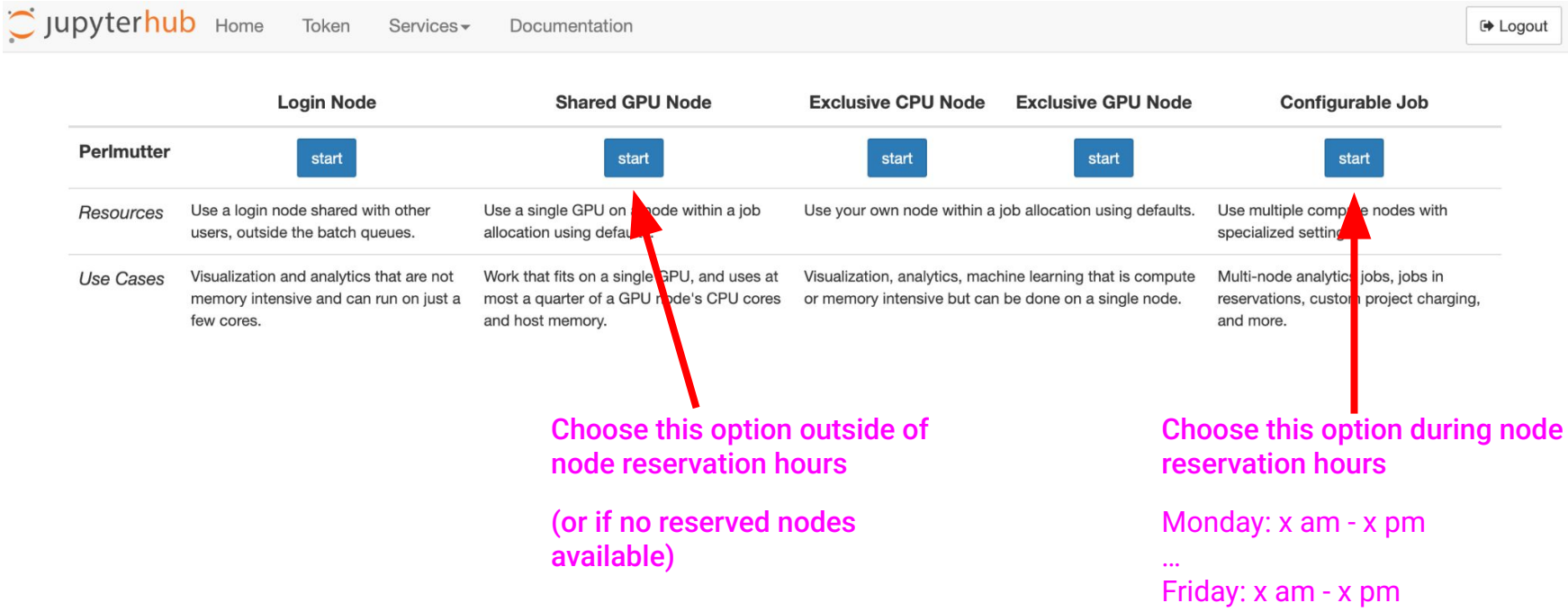
Remember to use the "Configurable Job" option during reservation hours (Central Time Zone)

Day	Reservation Name	Start Time	End Time
Jul 30	`intro_hpc_day0`	1 :00 PM	2:00 PM
Aug 11	`intro_hpc_day1`	2:30 PM	5:30 PM
Aug 12, AM	`intro_hpc_day2_am`	10:00 AM	11:30 AM
Aug 12, PM	`intro_hpc_day2_pm`	3:00 PM	5:30 PM
Aug 13	`intro_hpc_day3`	9:00 AM	5:30 PM
Aug 14	`intro_hpc_day4`	12:00 PM	8:30 PM

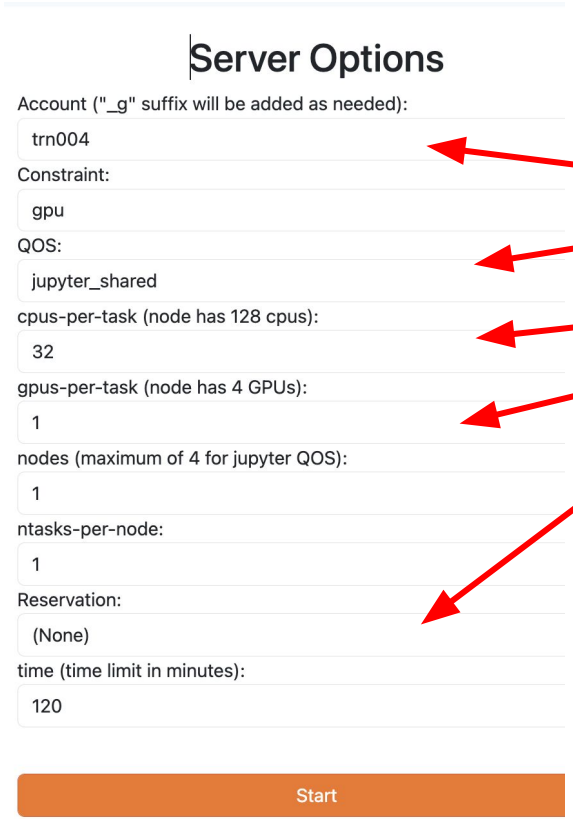
Outside of reservation hours, you can explore the other single-click options.

Hub Home Page or “Console”

All these options get you running Jupyter on Perlmutter but give you different ways to use its resources.



Running a Notebook Server Using a Single GPU



reservatio

Select **Account** : m4388

Select **QOS**: jupyter_shared

Lower **cpus-per-task** to: 32 (you can just type 32)

Lower **gpus-per-task** to: 1

Select **Reservation** : intro_hpc_day1

time (time limit in minutes): 120

Leave everything else the same

4 users will share one GPU node

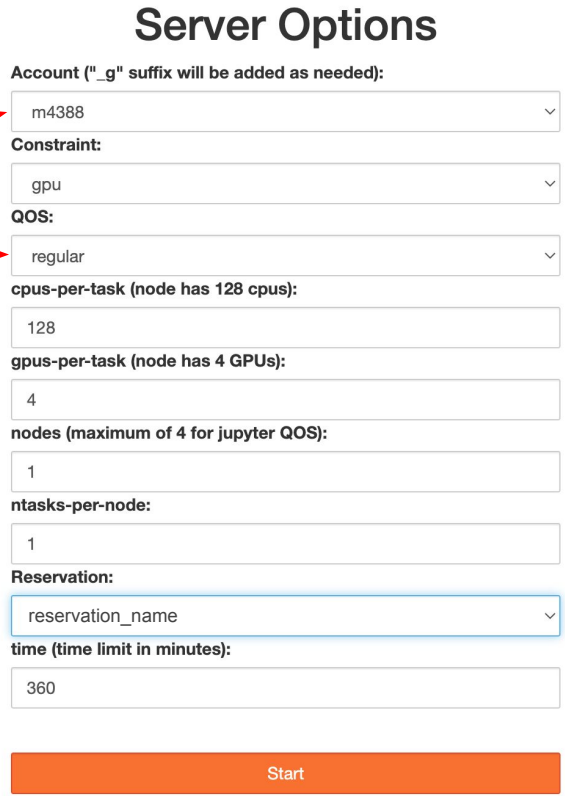
Running a Notebook Server Using One Whole Node

Select **Account** : m4388

Select **QOS**: regular

Select **Reservation** : ???

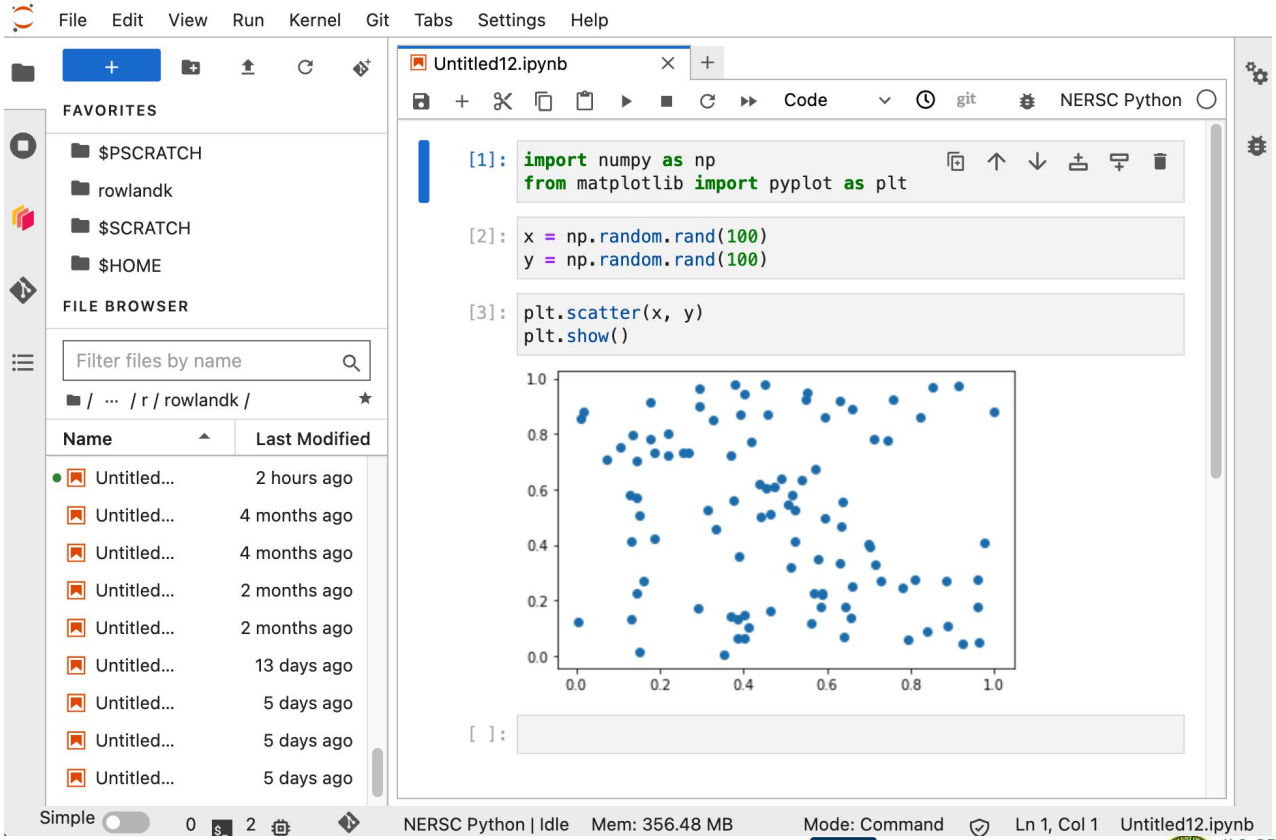
Leave everything else the same



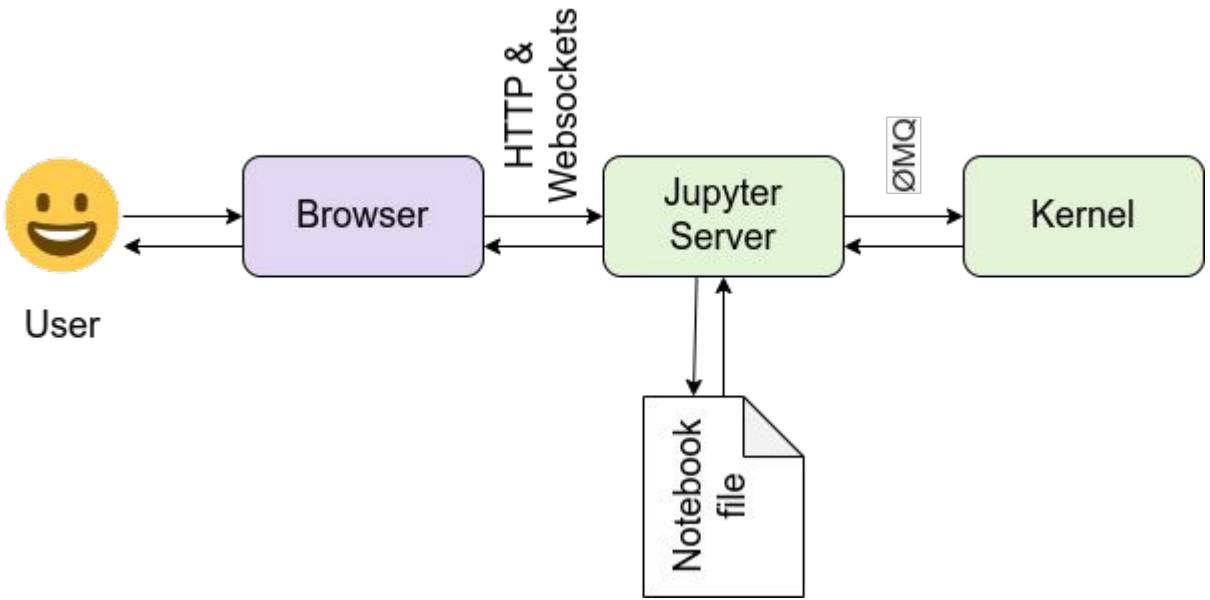
Use this setup if you want either:

- To use all 4 GPUs on a node, and/or
- All of the CPU cores on a node

JupyterLab Interface



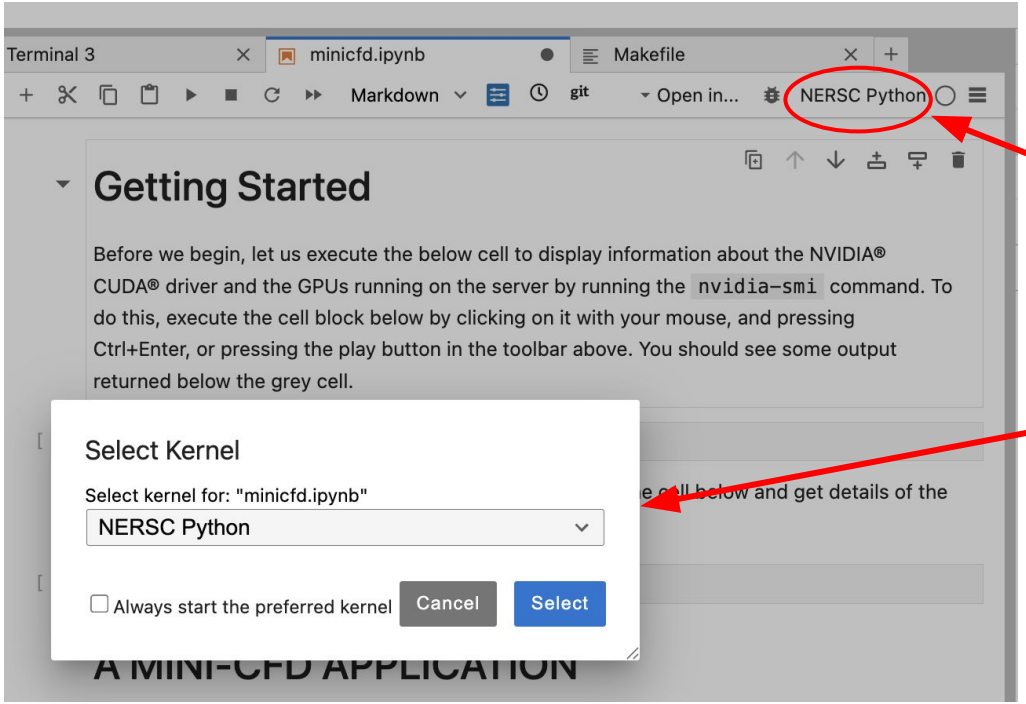
Kernels: How You Compute with Jupyter



- The kernel is what actually runs your code
- Default kernel is NERSC Python
  - From Python module
- Other kernels also provided
  - Julia
  - ML packages
- Bring your own kernel

https://docs.jupyter.org/en/latest/projects/architecture/content-architecture.html

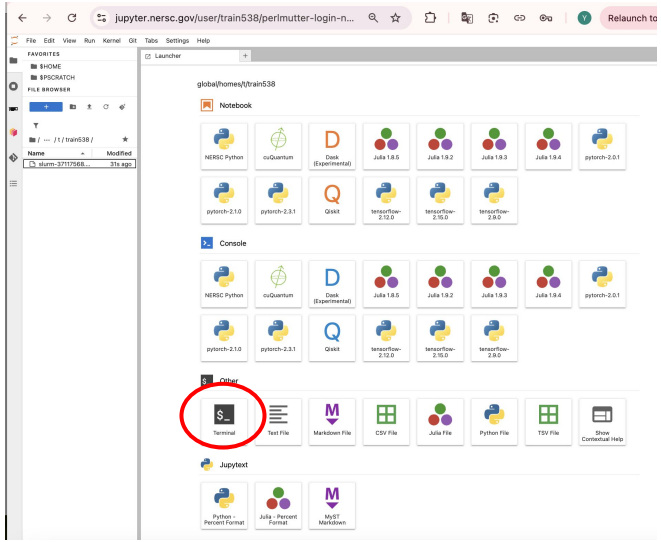
Selecting Your Notebook Kernel

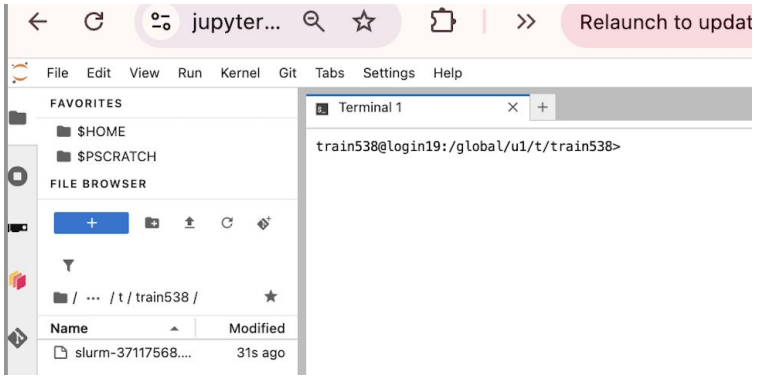


The default kernel is NERSC Python. Click on the kernel name and choose a different kernel from the drop down menu if needed.

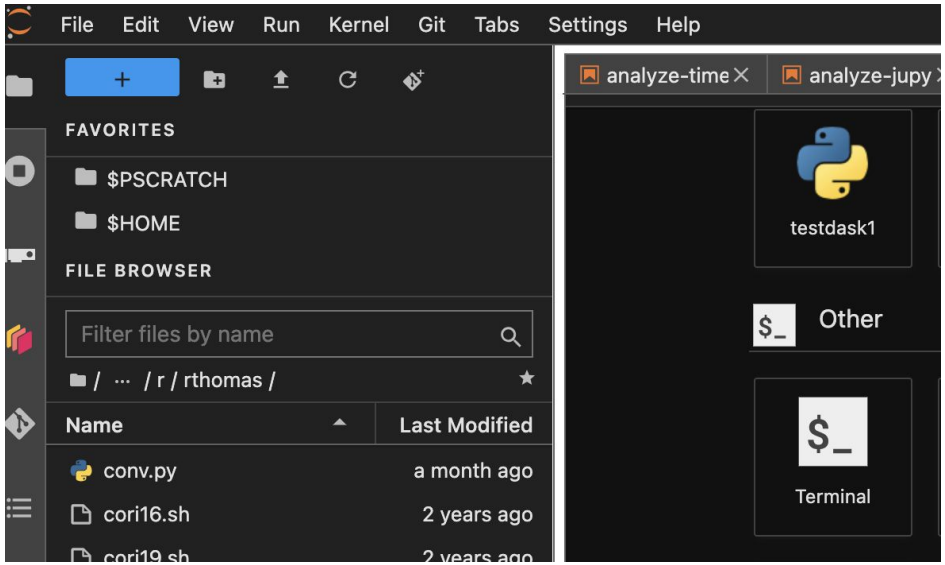
Using a Terminal in Jupyter

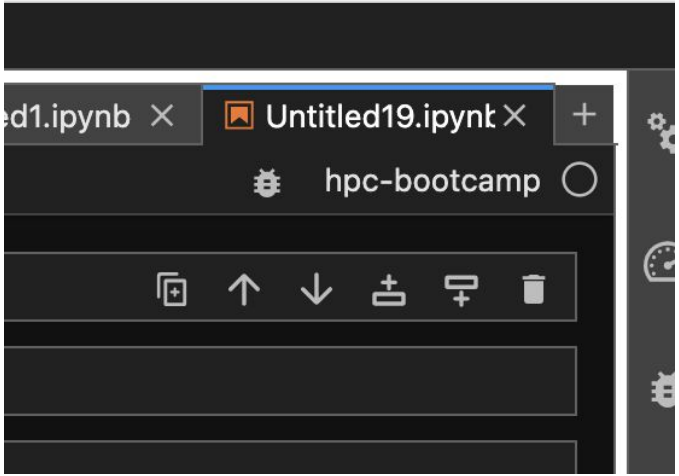
The Jupyter interface can be used to open a terminal prompt:





Demo if Time Allows





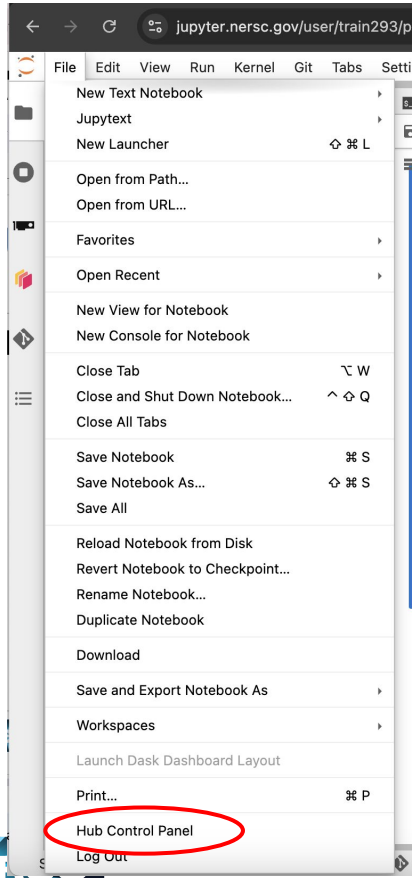
To open a terminal panel:

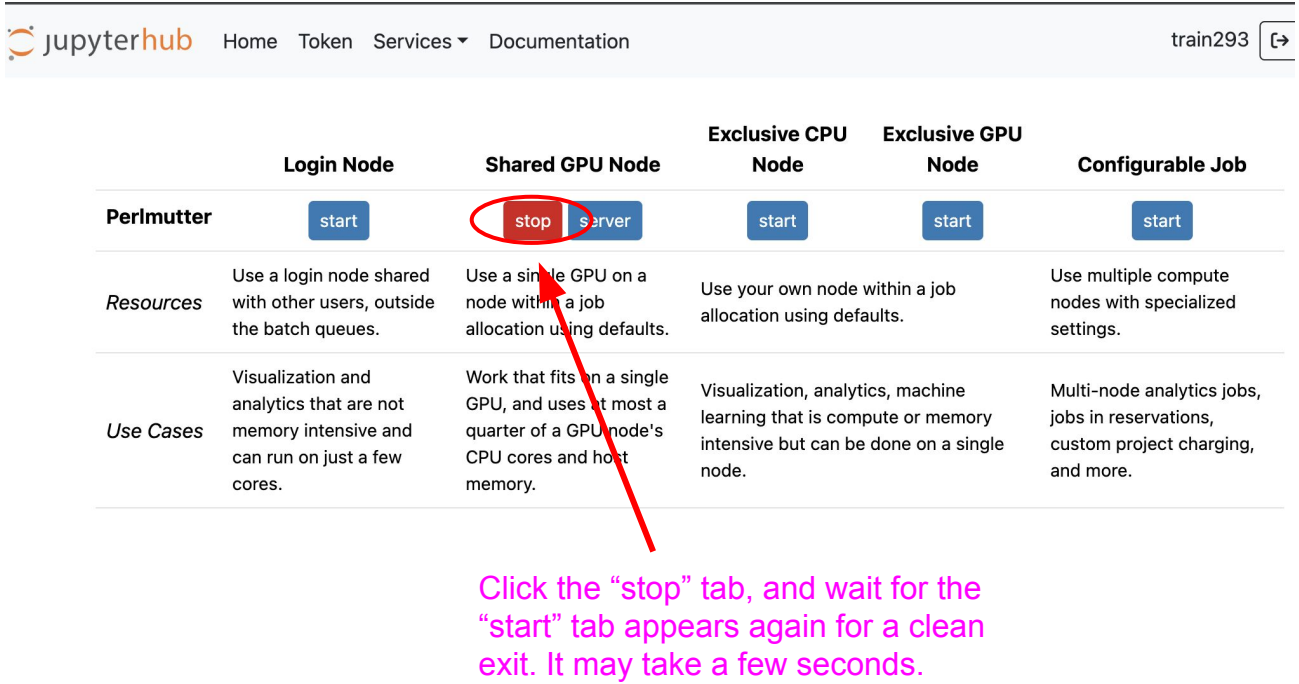
- Click the “+” in the top left corner
- Scroll down
- Select “Terminal” from under “Other”

Select your kernel at the top right

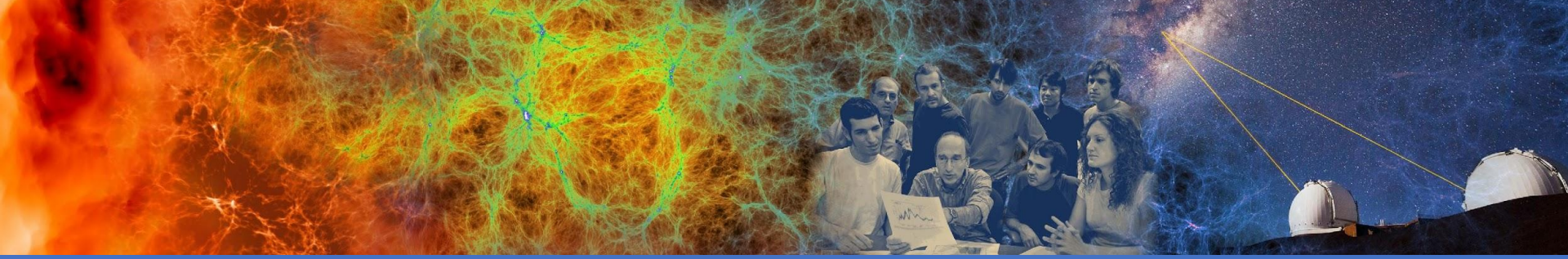
- Most projects can use “hpc-bootcamp”
- Several others you may want to use

How to Exit JupyterHub Cleanly





Click the “stop” tab, and wait for the “start” tab appears again for a clean exit. It may take a few seconds.



File Systems, Compile and Run Jobs

Transferring Data to Perlmutter

Great, now how do I get my files to onto the supercomputer?!

- For this workshop:
  - On NERSC internal filesystems (CFS): **`mv`**, **`cp`**, or **`rsync`**
    ```
    cp /path/to/original /path/to/new/copy
    ```
  - From your laptop: **`scp`**, **`rsync`**, drag and drop with Jupyter
    ```
    scp /path/on/laptop user@perlmutter.nersc.gov:/path/on/pm
    ```
  - From Github: **`git clone`**
    ```
    git clone https://www.github.com/ns/myrepo.git
    ```
- Other interesting use cases:
  - For large scientific data: Globus
  - When Globus doesn't work: **`rsync`**
  - Download from trusted URLs: **`wget`**, **`curl`**
  - Large, live, scientific data: come talk to us

Perlmutter File Systems

Global Home

- You land here when login
- Permanent, relatively small storage
- NOT tuned to perform well for parallel jobs
- Snapshot backups
- **Perfect for storing data such as source codes, shell scripts**
- **cd $HOME**

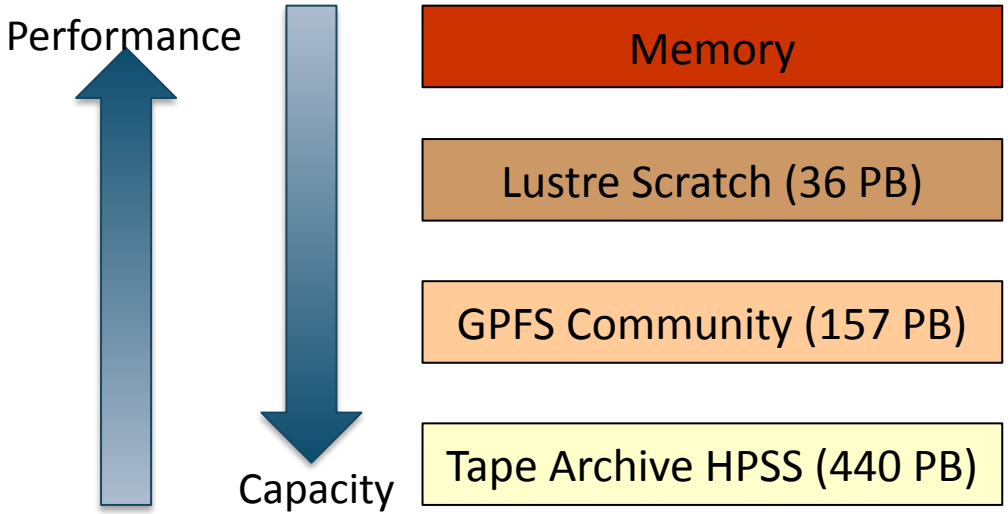
Community File System (CFS)

- Permanent, larger storage
- Medium performance for parallel jobs
- Snapshot backups
- **Perfect for sharing data within research group**
- **cd $CFS**

Scratch

- Large, temporary storage
- Optimized for read/write operations, NOT storage
- Not backed up
- Purge policy (8 weeks)
- **Perfect for staging data and performing computations**
- **cd $SCRATCH**

Simplified NERSC File Systems



Global Common (24 TB) — shared software

Global Home (408 TB) — individual home directories

Where should I work on my bootcamp project

- Materials for each project are available in the m4388 project area on Perlmutter CFS at **$CFS/m4388/Project***, such as **Project2**
- Students who work on each Group will work in a shared directory in **$CFS/m4388/Group***, such as **GroupC**

Where should I work on my bootcamp project

From a Terminal in JupyterHub

- To copy over the entire project to your group working directory
  - cd $CFS/m4388/GroupC
  - cp -r $CFS/m4388/Project2 . (notice the last dot)

    or: git clone https://github.com/<TBD>/intro-HPC-2025/Project2
- Any student could also do individual work in their own scratch directory
  - cd $SCRATCH
  - cp -r $CFS/m4388/Project2 . (notice the last dot)

    or: git clone https://github.com/<TBD>/intro-HPC-2025/Project2

Programming Environment and Compile

- Some users use JupyterHub to login and mostly using Python for data analytics. There is a “terminal” kernel in JupyterHub.
- Most users also directly login to Perlmutter with SSH from a terminal, and work on scientific applications written in C/C++ and Fortran
  - These codes need to be compiled first, then run the generated executable on compute nodes
- There are multiple compilers available on Perlmutter
  - The default is GCC compiler
- Compiler wrappers are used to compile, such as
  - cc -o mycode.exe mycode.c
  - CC -o mycode.exe mycode.cc
  - ftn -o mycode.exe mycode.f90

Jobs at NERSC

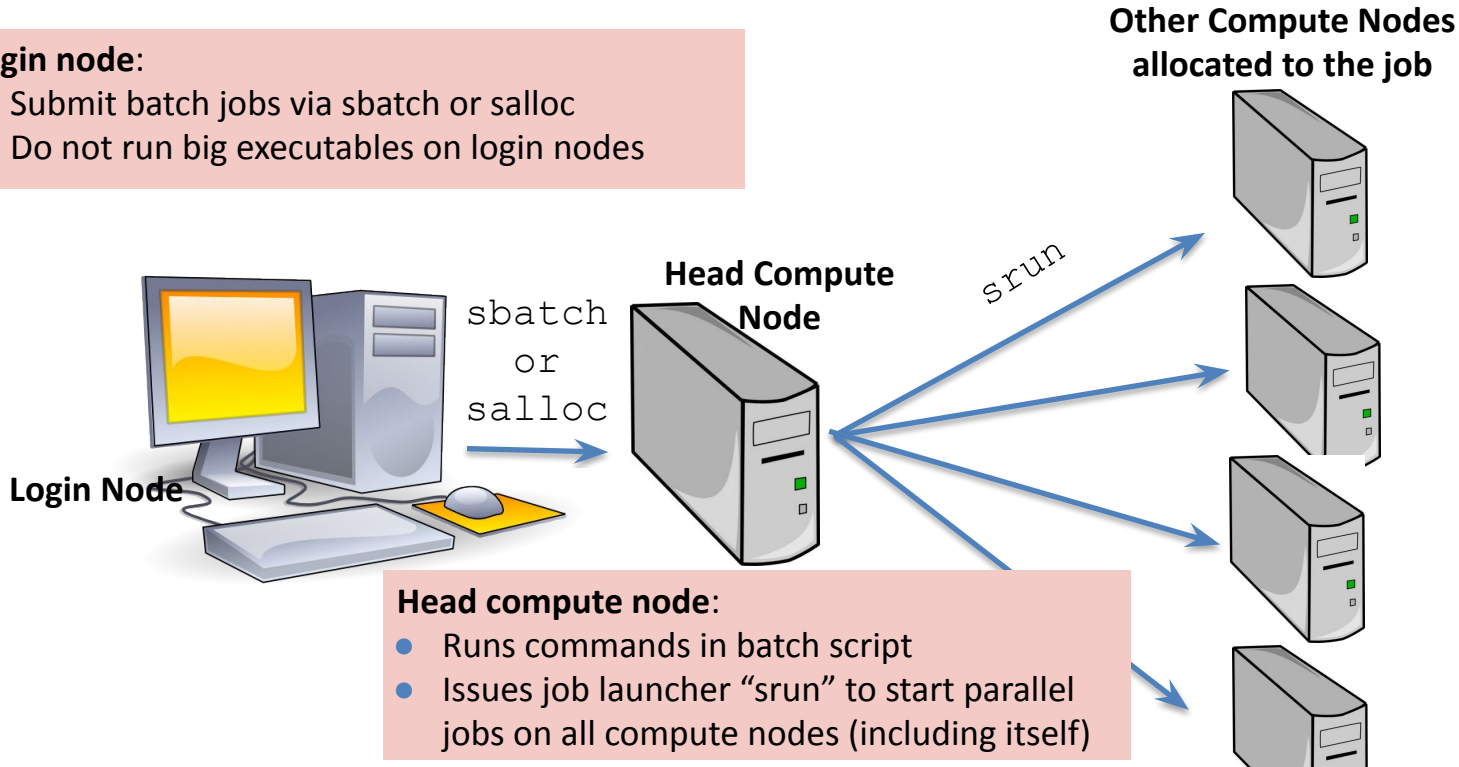
- Most are **parallel jobs** (10s to 100,000+ cores)
  - Meaning a job is run with multiple MPI tasks, each task tackle a subproblem, such as a subdomain
- Also a number of **“serial” jobs**
  - Typically “pleasantly parallel” simulation or data analysis
- Production runs execute in batch mode
- Our batch scheduler is SLURM
- Typical run times are a few to 10s of hours
  - Limits are necessary because of MTBF and the need to accommodate 9,000 users’ jobs

Login Nodes and Compute Nodes

- Login nodes
  - Edit files, compile codes, submit batch jobs, etc.
  - Run short, serial utilities and applications
- Compute nodes
  - Execute your application
  - Dedicated resources for your job
  - Perlmutter has CPU and GPU compute nodes

Launching Parallel Jobs with Slurm

Login node:
- Submit batch jobs via sbatch or salloc
- Do not run big executables on login nodes



Head compute node:
- Runs commands in batch script
- Issues job launcher "srun" to start parallel jobs on all compute nodes (including itself)

My First “Hello World” Program

```
/* C Example, mpi-hello.c */
#include <stdio.h>
#include <mpi.h>

int main (argc, argv)
     int argc;
     char *argv[];
{
  int rank, size;

  MPI_Init (&argc, &argv);    /* starts MPI */
  MPI_Comm_rank (MPI_COMM_WORLD, &rank);     /* get current process id */
  MPI_Comm_size (MPI_COMM_WORLD, &size);     /* get number of processes */
  printf( "Hello world from process %d of %d\n", rank, size);
  MPI_Finalize();
  return 0;
}
```

To compile:
% cc -o mpi-hello mpi-hello.c

Run “Hello World” Program

```
my_batch_script:
(request 2 CPU nodes for 10 min, run in debug queue)

#!/bin/bash
#SBATCH -q debug
#SBATCH -N 2
#SBATCH -t 10:00
#SBATCH -C cpu

# run with 8 MPI tasks (this is a comment)
srun -n 8 -c 64 –cpu-bind=cores ./mpi-hello
```

To run via batch queue
% sbatch submit_job.sh
To run via interactive batch
login% salloc -N 2 -q interactive -C cpu -t 10:00
<Wait for session prompt. Land on a compute node>
compute% **srun -n 8** -c 64 –cpu-bind=cores **./mpi-hello**

Run with GPU Node Reservation in Shared

```
my_batch_script:
(request 2 CPU nodes for 10 min, run in debug queue)

#!/bin/bash
#SBATCH -N 2
#SBATCH -t 10:00
#SBATCH -A cpu
#SBATCH -m4388
#SBATCH -q shared
#SBATCH --reservation=intro_hpc_day0
# run with 8 MPI tasks (this is a comment)
srun -n 8 -c 64 –cpu-bind=cores ./mpi-hello
```

To run via batch queue

% sbatch submit_job_res.sh

To run via interactive batch

login% salloc -N 2 -q interactive -C gpu -A m4388 -q shared --reservation=intro_hpc_day0 -t 10:00

<Wait for session prompt. Land on a compute node>

compute% **srun -n 8** -c 64 –cpu-bind=cores **./mpi-hello**

Monitor Your Batch Jobs

- squeue
  - By default squeue displays jobs from all users
- sqs
  - sqs is a NERSC wrapper on squeue
  - By default sqs displays jobs from current user

Compile and Run Demo

Commands Used in Compile and Run Demo

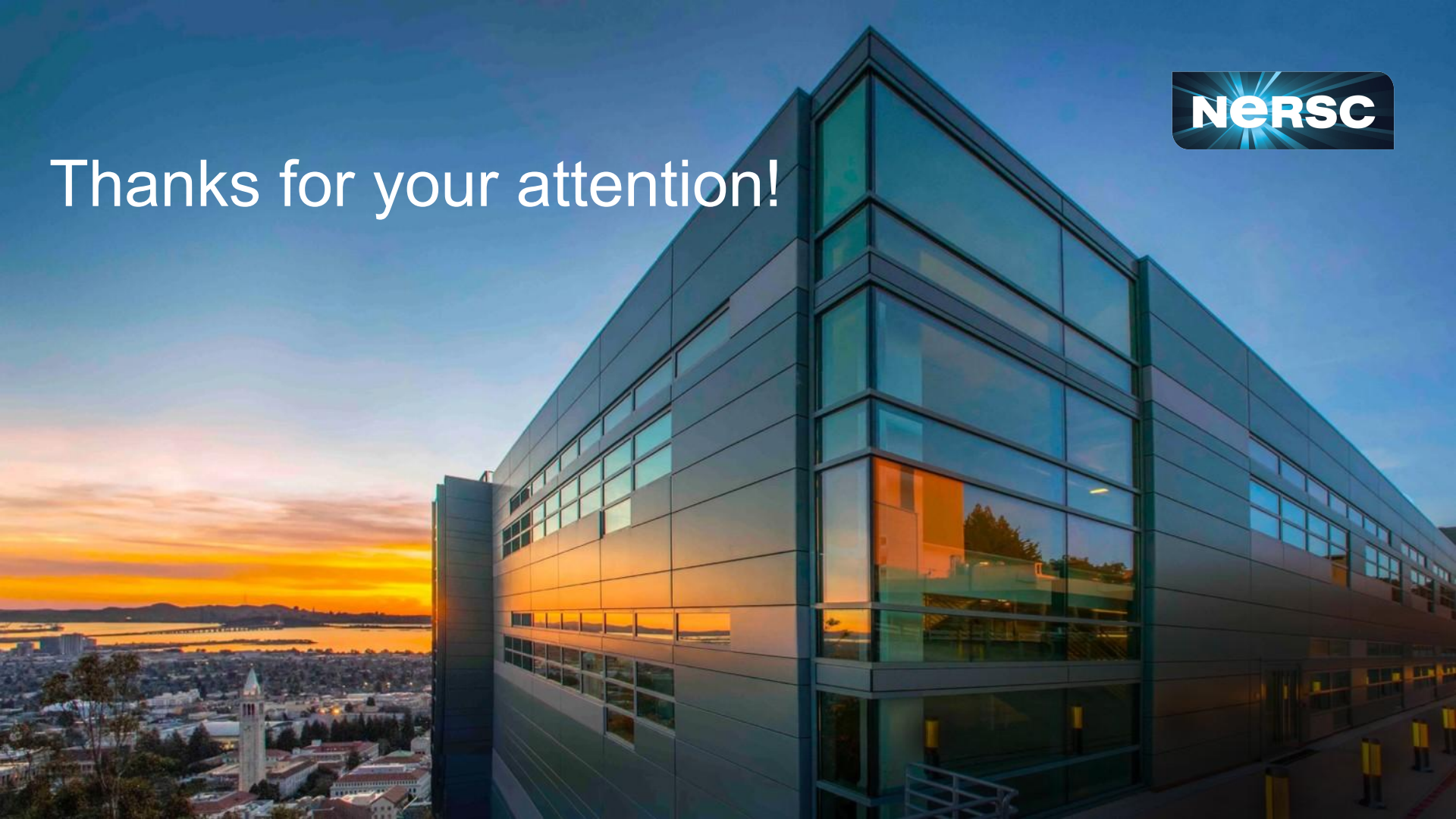
```
% pwd
% cd $SCRATCH
% cp -r $CFS/m4388/sample_compile_run .
% cd sample_compile_run
% ls
  mpi-hello.c mpi-hello.cc  mpi-hello.f90  submit_job.sh
% more mpi-hello.c
% cc -o mpi-hello mpi-hello.c
(or % CC -o mpi-hello mpi-hello.cc
 or % ftn -o mpi-hello mpi-hello.f90)
% more submit_job.sh
% sbatch submit_job.sh
% sqs
% squeue |more
% more slurm-*.out
% salloc -N 2 -C cpu -t 10:00 -q interactive
  <wait for allocation>
  % srun -n 8 -c 64 --cpu-bind=cores ./mpi-hello
```

Using GPU Node Reservations in Shared

```
% more submit_job_res.sh
% sbatch submit_job_res.sh
% sqs
% squeue |more
% more slurm-*.out
% salloc -N 2 -C gpu -A m4388 -q shared --reservation=intro_hpc_day0
-t 10:00 -q interactive
  <wait for allocation>
% srun -n 8 -c 64 --cpu-bind=cores ./mpi-hello
```

If You Have Any Questions

- Short Term
  - Office Hours next week, Aug 6, 10-11 am Pacific
  - Slack channel: TBA
  - Ask trainers, peer mentors, group members for help now through Bootcamp
- Longer term (your NERSC account is valid through 01/15/2025):
  - Join NERSC user Slack channel
  - Submit a ticket via NERSC Help Portal
  - Check NERSC Docs: https://docs.nersc.gov/


NERSC
Thanks for your attention!