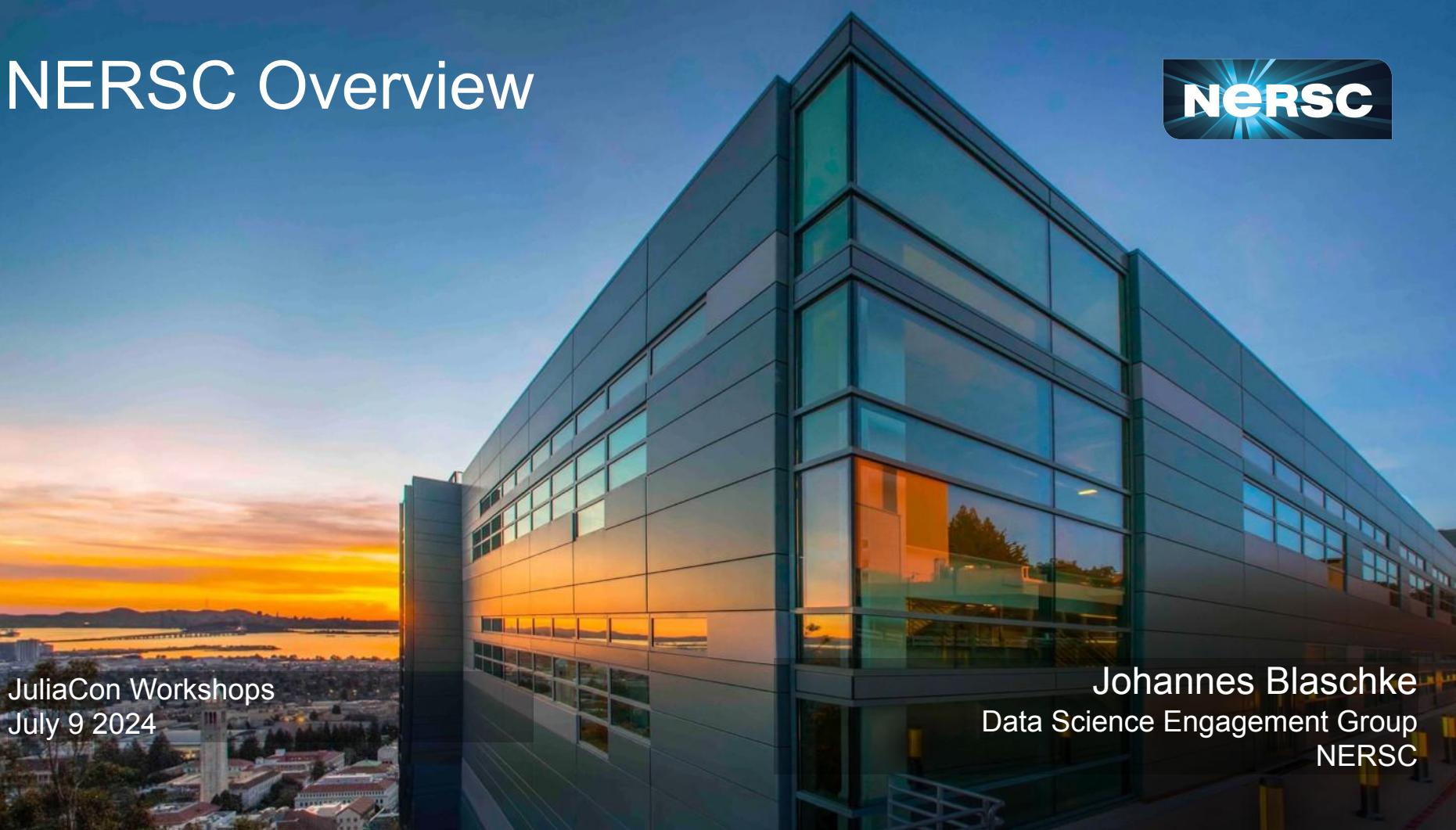


NERSC Overview



JuliaCon Workshops
July 9 2024

Johannes Blaschke
Data Science Engagement Group
NERSC

National Energy Research Scientific Computing Center

- NERSC is a national supercomputer center funded by the U.S. Department of Energy Office of Science (SC)
 - Supports SC research mission
 - Part of Berkeley Lab
- Researchers with funding from SC who need supercomputing resources can use NERSC
 - Other researchers can apply if research is in SC mission
- NERSC supports 10,000 users, 1,000 projects
 - From all 50 states + international; 60% from universities
 - Hundreds of users log on each day

NERSC User Demographics

~10,000 Annual Users from ~800 Institutions + National Labs



32%
Graduate
Students



19%
Postdoctoral
Fellows



15%
Staff
Scientists



13%
University
Faculty



8%
Undergraduate
Students



5%
Professional
Staff



60%
Universities



29%
DOE Labs



5%
Other
Government Labs



4%
Industry

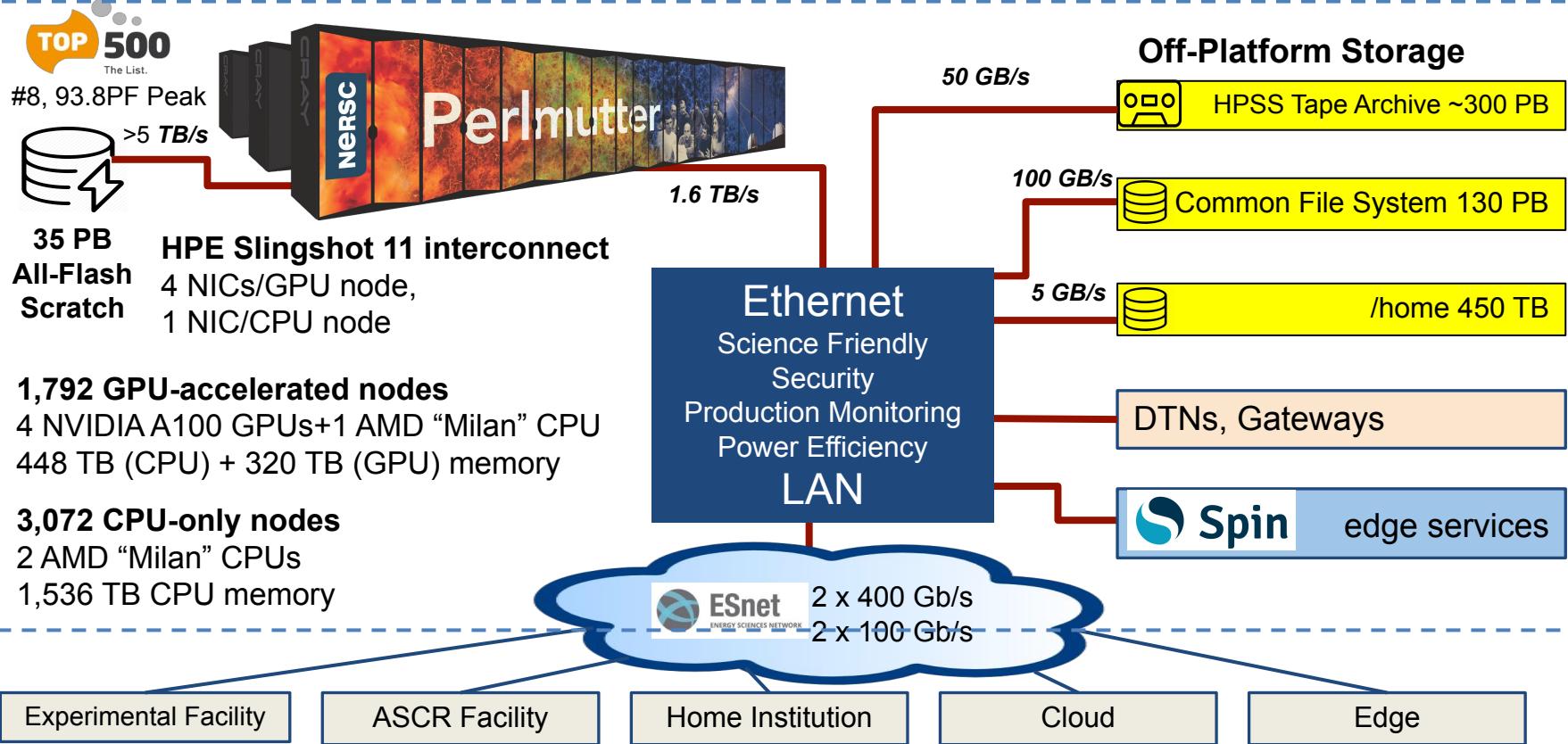


1%
Small
Businesses



<1%
Private Labs

NERSC Systems Ecosystem



Perlmutter system configuration

NVIDIA "Ampere" GPU Nodes

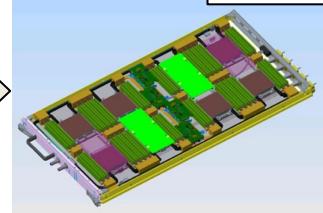
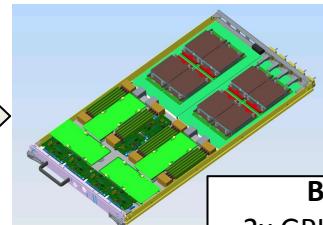
4x GPU + 1x CPU
40 GiB HBM + 256 GiB DDR
4x 200G "Slingshot" NICs

AMD "Milan" CPU Node

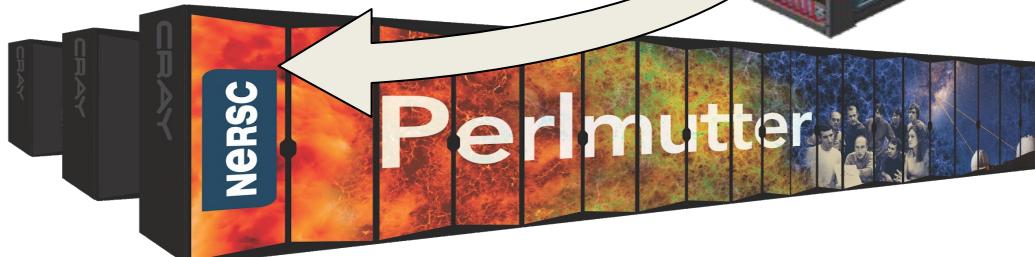
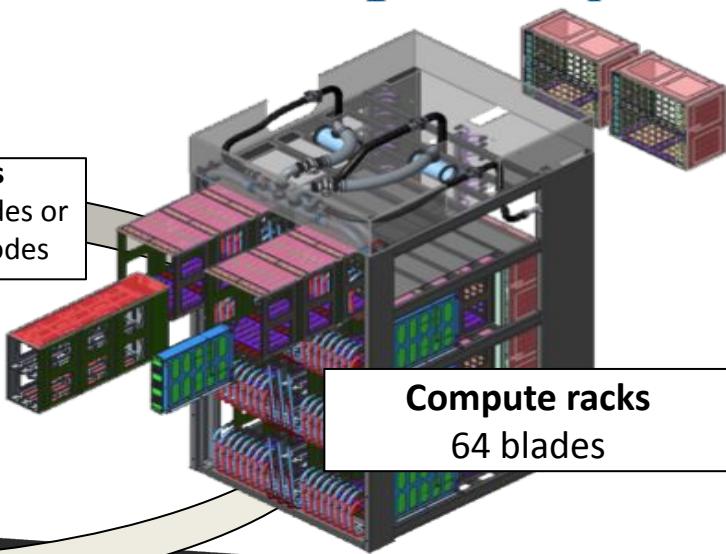
2x CPUs
> 256 GiB DDR4
1x 200G "Slingshot" NIC

Centers of Excellence
Network
Storage
App. Readiness
System SW

Perlmutter system
GPU racks
CPU racks
~6 MW



Blades
2x GPU nodes or
4x CPU nodes

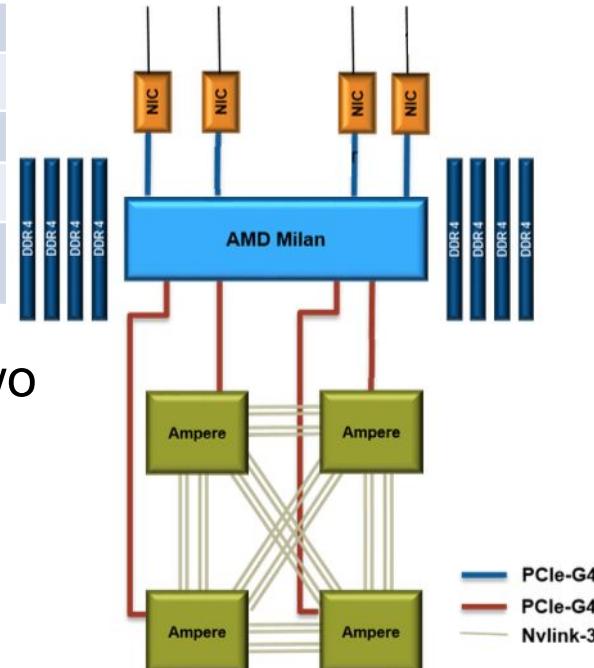


Node Types

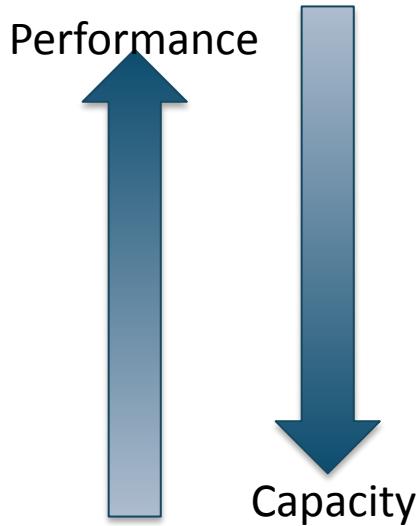
Partition	Nodes	CPU	RAM	GPU	NIC
GPU	1536	1x AMD EPYC 7763	256GB	4x NVIDIA A100 (40GB)	4x HPE Slingshot 11
	256	1x AMD EPYC 7763	256GB	4x NVIDIA A100 (80GB)	4x HPE Slingshot 11
CPU	3072	2x AMD EPYC 7763	512GB	–	1x HPE Slingshot 11
Login	40	1x AMD EPYC 7713	512GB	4x NVIDIA A100 (40GB)	–
Large Memory	4	1x AMD EPYC 7713	1TB	4x NVIDIA A100 (40GB)	1x HPE Slingshot 11

- Each Milan CPU has 64 2.45 GHz cores with two hardware threads
- Access is managed via Slurm partitions
- Login nodes are GPU-enabled

GPU Node Architecture:



Simplified NERSC File Systems



Memory

Burst Buffer

Scratch

Community

HPSS

Global Common

Global Home

35 PB Flash Scratch

Lustre >5 TB/s
temporarily (purge)

xGB (x=Total RAM) In-Memory Burst Buffer

/tmp RamFS

28 PB (Cori) HDD Scratch

Lustre 700 GB/s,
temporary (12 wk purge)

157 PB HDD Community

Spectrum Scale (GPFS)
150 GB/s, permanent

150 PB Tape Archive

HPSS Forever

20 TB SSD Software

Spectrum Scale
Permanent
Faster compiling / Source Code

Global File Systems

Global Home

- Permanent, relatively small storage
- Mounted on all platforms
- NOT tuned to perform well for parallel jobs
- Quota cannot be changed
- Snapshot backups (7-day history)
- **Perfect for storing data such as source code, shell scripts**
- **Addressed using \$HOME**

Community File System (CFS)

- Permanent, larger storage
- Mounted on all platforms
- Medium performance for parallel jobs
- Quota can be changed
- Snapshot backups (7-day history)
- **Perfect for sharing data within research group**
- **Addressed using \$CFS**

Local File Systems

Scratch

- Large, temporary storage
- Optimized for read/write operations, NOT storage
- Not backed up
- Purge policy (12 weeks)
- **Perfect for staging data and performing computations**
- **Addressed using \$SCRATCH**

Burst Buffer

- Temporary storage
- High-performance in-memory file system
- **Perfect for getting good performance in I/O-constrained codes**
- **Not shared between nodes**

There are many different ways to access NERSC. To use our resources, you need to either:

1. Log into the login nodes and interact with Slurm (covered here)
2. Use services (eg. Jupyter) that expose web interfaces (covered later)
3. Interact via our REST API – called the “Superfacility API”
(covered later)

Connecting with SSH

- To access Perlmutter, use:

saul-p1.nersc.gov

perlmutter-p1.nersc.gov



```
blaschke@laptop:~$ ssh <user>@saul-p1.nersc.gov  
blaschke@laptop:~$ ssh <user>@perlmutter-p1.nersc.gov
```

- To be able to open GUI applications on the login nodes
use: ssh -Y



```
blaschke@laptop:~$ ssh -Y <user>@saul-p1.nersc.gov
```

Connecting with SSH

After successfully logging on, you will be greeted by the terms of use, and the command-line input prompt:

For past outages, see: <https://my.nersc.gov/outagelog-cs.php>
blaschke@perlmutter:login17: ~

From here you can interact with
perlmutter...

12

Submitting Jobs

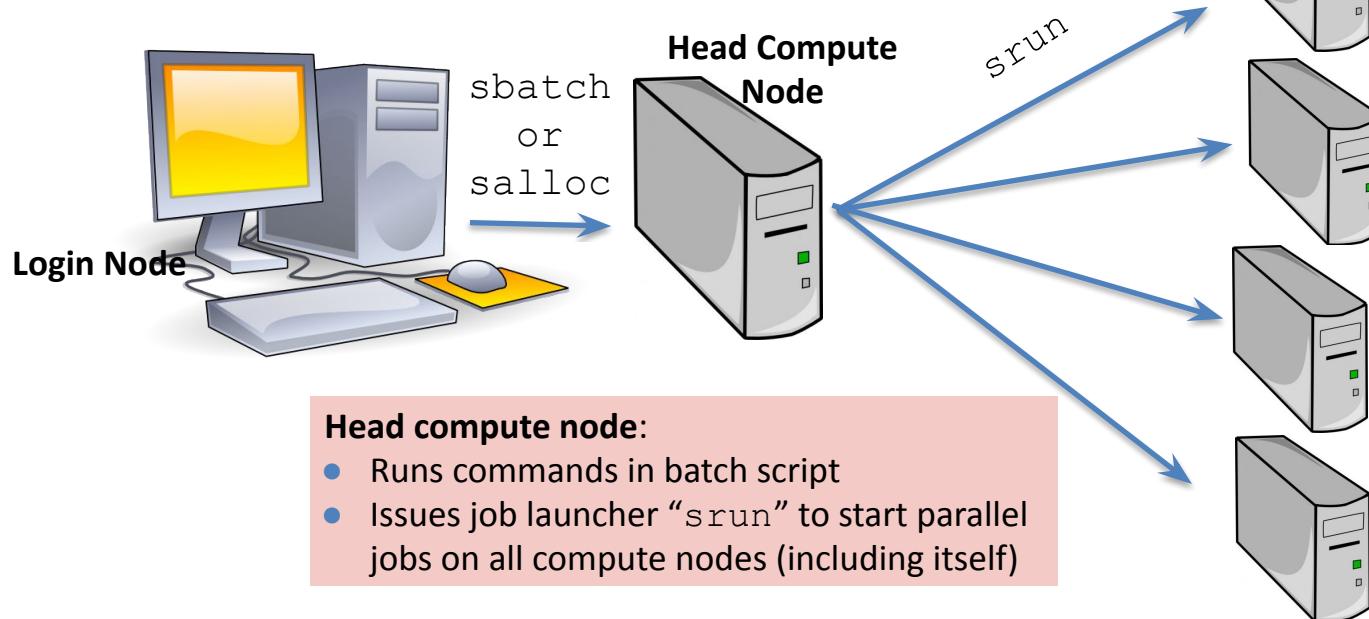
- Jobs can be submitted to queueing system through sbatch or salloc:
 - `sbatch <my_job_script>`
 - `salloc <options>`
- The above methods list details about resources needed for a job and for how long
- Eg.: a request for a cpu node to be allocated for 5 mins:
 - `salloc -N 1 -C cpu -q debug -t 5 -A <project>`



Submitting Jobs

Login node:

- Submit batch jobs via `sbatch` or `salloc`



*figure courtesy Helen (2020 NERSC Training)

My First “Hello World” Job Script

debug queue

2 nodes

10 min “walltime”

run on haswell partition

use SCRATCH file system



my_batch_script.sh

```
#!/bin/bash
#SBATCH -q debug
#SBATCH -N 2
#SBATCH -t 10:00
#SBATCH -C haswell
#SBATCH -L SCRATCH
#SBATCH -J myjob
srun -n 64 ./helloWorld
```

To run via batch queue

```
% sbatch my_batch_script.sh
```

To run via interactive batch

```
% salloc -N 2 -q interactive -C haswell -t 10:00
```

<wait_for_session_prompt. Land on a compute node>

```
% srun -n 64 ./helloWorld
```

Anatomy of a Batch Script



my_batch_script.sh

```
#!/bin/bash
#SBATCH --account=mxxx
#SBATCH --qos=regular
#SBATCH --nodes=2
#SBATCH --time=60
#SBATCH --constraint=gpu
#SBATCH --jobname=myjob
#SBATCH --license=scratch,cfs

export OMP_NUM_THREADS=1
srun -n 64 <executable>
```



my_batch_script.sh

```
#!/bin/bash
#SBATCH -A mxxx
#SBATCH -q regular
#SBATCH -N 2
#SBATCH -t 12:00:00
#SBATCH -C gpu
#SBATCH -J myjob
#SBATCH -L scratch,cfs

export OMP_NUM_THREADS=1
srun -n 64 <executable>
```

Anatomy of a Batch Script

! – Job scripts are bash scripts



my_batch_script.sh

```
#!/bin/bash
#SBATCH --account=xxxx
#SBATCH --qos=regular
#SBATCH --nodes=2
#SBATCH --time=60
#SBATCH --constraint=gpu
#SBATCH --jobname=myjob
#SBATCH --license=scratch,cfs

export OMP_NUM_THREADS=1
srun -n 64 <executable>
```



my_batch_script.sh

```
#!/bin/bash
#SBATCH -A xxxx
#SBATCH -q regular
#SBATCH -N 2
#SBATCH -t 12:00:00
#SBATCH -C gpu
#SBATCH -J myjob
#SBATCH -L scratch,cfs

export OMP_NUM_THREADS=1
srun -n 64 <executable>
```

Anatomy of a Batch Script

Preamble. All flags after `#SBATCH` are passed to slurm



my_batch_script.sh

```
#!/bin/bash
#SBATCH --account=xxxx
#SBATCH --qos=regular
#SBATCH --nodes=2
#SBATCH --time=60
#SBATCH --constraint=gpu
#SBATCH --jobname=myjob
#SBATCH --license=scratch,cfs
```

```
export OMP_NUM_THREADS=1
srun -n 64 <executable>
```



my_batch_script.sh

```
#!/bin/bash
#SBATCH -A xxxx
#SBATCH -q regular
#SBATCH -N 2
#SBATCH -t 12:00:00
#SBATCH -C gpu
#SBATCH -J myjob
#SBATCH -L scratch,cfs
```

```
export OMP_NUM_THREADS=1
srun -n 64 <executable>
```

Anatomy of a Batch Script

Setup. Environment variables control many aspects of HPC libraries.



my_batch_script.sh

```
#!/bin/bash
#SBATCH --account=xxxx
#SBATCH --qos=regular
#SBATCH --nodes=2
#SBATCH --time=60
#SBATCH --constraint=gpu
#SBATCH --jobname=myjob
#SBATCH --license=scratch,cfs

export OMP_NUM_THREADS=1
srun -n 64 <executable>
```



my_batch_script.sh

```
#!/bin/bash
#SBATCH -A xxxx
#SBATCH -q regular
#SBATCH -N 2
#SBATCH -t 12:00:00
#SBATCH -C gpu
#SBATCH -J myjob
#SBATCH -L scratch,cfs

export OMP_NUM_THREADS=1
srun -n 64 <executable>
```

Anatomy of a Batch Script

srun launches the executable on all nodes (and provides MPI)



my_batch_script.sh

```
#!/bin/bash
#SBATCH --account=xxxx
#SBATCH --qos=regular
#SBATCH --nodes=2
#SBATCH --time=60
#SBATCH --constraint=gpu
#SBATCH --jobname=myjob
#SBATCH --license=scratch,cfs

export OMP_NUM_THREADS=1
srun -n 64 <executable>
```



my_batch_script.sh

```
#!/bin/bash
#SBATCH -A xxxx
#SBATCH -q regular
#SBATCH -N 2
#SBATCH -t 12:00:00
#SBATCH -C gpu
#SBATCH -J myjob
#SBATCH -L scratch,cfs

export OMP_NUM_THREADS=1
srun -n 64 <executable>
```

Sample Batch Script for a GPU node



gpu_node.sh

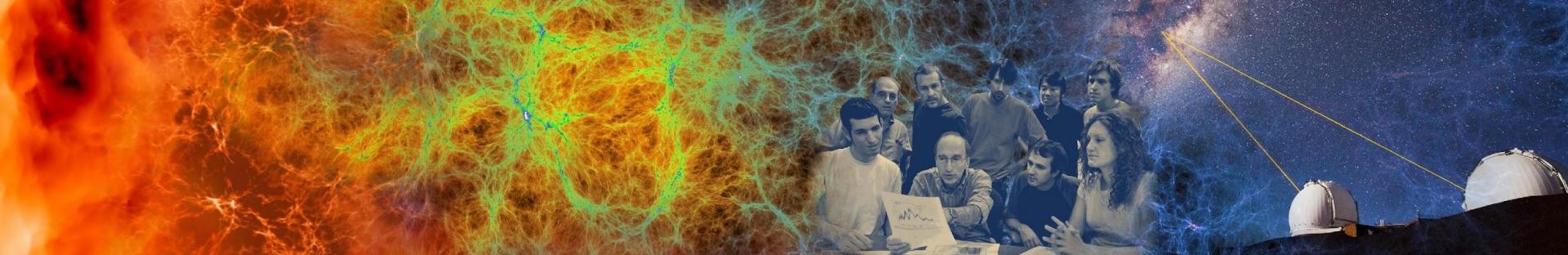
```
#!/bin/bash
#SBATCH --account=mxxx
#SBATCH --qos=regular
#SBATCH --nodes=2
#SBATCH --time=60
#SBATCH --constraint=gpu
#SBATCH --jobname=myjob
#SBATCH --ntasks-per-node=64
#SBATCH --cpus-per-task=2
#SBATCH --gpus-per-node=4
```

```
export OMP_NUM_THREADS=1
srun -n 128 <executable>
```

$$c = 2 * (64/k)$$

where:

k = ntasks-per-node



NERSC Training Accounts

for *Hands-on with Julia for HPC on GPUs and CPUs*



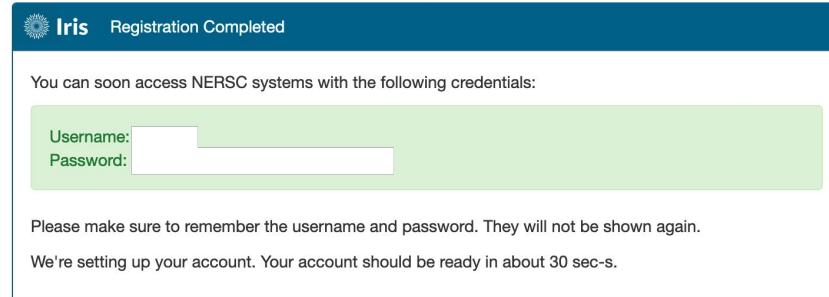
BERKELEY LAB



Office of
Science

Activate your Training Account

- Go to: <https://iris.nersc.gov/train>
- Enter the Training Code: “enxl” along with your details
Note: if you have previously registered an account using a different code please recreate it.
- **Important: make a copy of your login and password**
you won’t be able to change these, nor recover them later!!!



Access to Perlmutter and Use Julia Module

- NERSC users have been added to ntrain9 project
- Non-users were sent the instruction to get a training account
 - Account valid through July 14
- Login to Perlmutter: ssh username@perlmutter.nersc.gov
- Julia modules:
 - % module use /global/common/software/nersc/n9/julia/modules
 - % module load PrgEnv-gnu
 - % module load julia
- Running Jobs examples:
 - <https://docs.nersc.gov/jobs/>

Compute Node Reservations

- GPU node reservation: 1pm - 4pm EDT, Tues June 18
 - To use 1 GPU only (sample flags for sbatch or salloc):
 - `-A ntrain1 --reservation=julia_training -C gpu -N 1 -c 32 -G 1 -t 30:00 -q shared`
 - To use multiple nodes (sample flags for sbatch or salloc):
 - `-A ntrain1 --reservation=julia_hpc_18 -C gpu -N 2 -t 30:00 -q regular`
- Outside of reservation, use:
 - To use 1 GPU only (sample flags for sbatch or salloc):
 - `-A <project> -C gpu -N 1 -c 32 -G 1 -t 30:00 -q shared`
 - To use multiple nodes (sample flags for sbatch or salloc):
 - `-A <project> -C gpu -N 2 -t 30:00 -q regular (or -q interactive for salloc)`

Reminder about file Systems

`$HOME` is not the best file system for running jobs at scale.
Consider therefore replacing all occurrences of `$HOME` with
`$SCRATCH` in the NERSC scripts here:

<https://github.com/JuliaHPC/juliacon24-hpcworkshop.git>

Pro tip: add `export JULIA_DEPOT_PATH=$SCRATCH/depot`
to your scripts for optimal performance (remember to do this in
the config script, **and** the job scrip)



Logging into Jupyter

- Go to <https://jupyter.nersc.gov>
- Sign in using your training account credentials
- Select your preferred jupyter instance:

The screenshot shows the jupyterhub login page. At the top, there is a navigation bar with the jupyterhub logo, Home, Token, Services (with a dropdown arrow), Documentation, a user ID train130, and a Logout button. Below the navigation bar, a yellow banner displays a message about NERSC resources being in maintenance or having issues, and it links to the NERSC MOTD for further information. It also mentions the Perlmutter status as degraded. The main content area contains a table comparing four types of nodes: Shared CPU Node, Exclusive CPU Node, Exclusive GPU Node, and Configurable GPU. Each row in the table corresponds to a specific NERSC system (Perlmutter, Cori) and provides a 'start' button. The table includes sections for Resources and Use Cases.

	Shared CPU Node	Exclusive CPU Node	Exclusive GPU Node	Configurable GPU
Perlmutter	<button>start</button>	<button>start</button>	<button>start</button>	<button>start</button>
Cori	<button>start</button>			<button>start</button>
Resources	Use a node shared with other users' notebooks but outside the batch queues.	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.	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.	



Setup

- Go to <https://jupyter.nersc.gov> and select:
“Exclusive GPU Node” in the “Perlmutter” row

The screenshot shows the jupyterhub interface with a navigation bar at the top. Below the navigation bar, there is a yellow banner with the following message: "The following NERSC resources that Jupyter depends on appear to be in maintenance or having issues. This may impact Jupyter. See the [NERSC MOTD](#) for further information. Perlmutter status: degraded". The main content area contains a table with four columns: "Shared CPU Node", "Exclusive CPU Node", "Exclusive GPU Node", and "Configurable GPU". The "Exclusive GPU Node" column is highlighted with a red box around the "start" button for the "Perlmutter" row. The table rows are as follows:

	Shared CPU Node	Exclusive CPU Node	Exclusive GPU Node	Configurable GPU
Perlmutter	start	start	start	start
Cori	start			start
Resources	Use a node shared with other users' notebooks but outside the batch queues.	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.	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.



Setup

If you're spending a long time in the queue waiting for an exclusive GPU node, "Shared GPU Node"s are also available – eg:

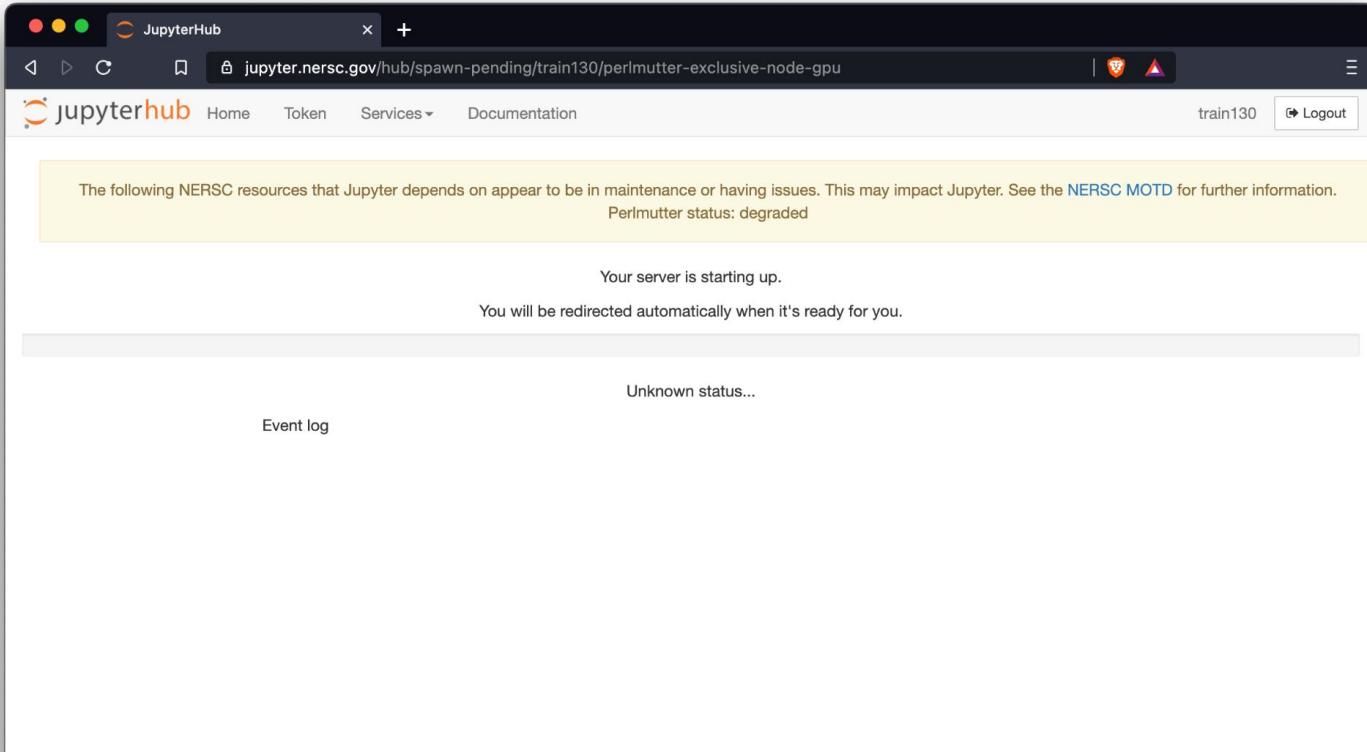
	Login Node	Shared GPU Node	Exclusive CPU Node	Exclusive GPU Node	Configurable Job
Alvarez	<button>start</button>	<button>start</button>	<button>start</button>	<button>start</button>	
Muller	<button>start</button>	<button>start</button>	<button>start</button>	<button>start</button>	<button>start</button>
Perlmutter	<button>stop</button> <button>server</button>	<button>start</button>	<button>start</button>	<button>start</button>	<button>start</button>
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.	

Alternatively please use a reservation (details later)



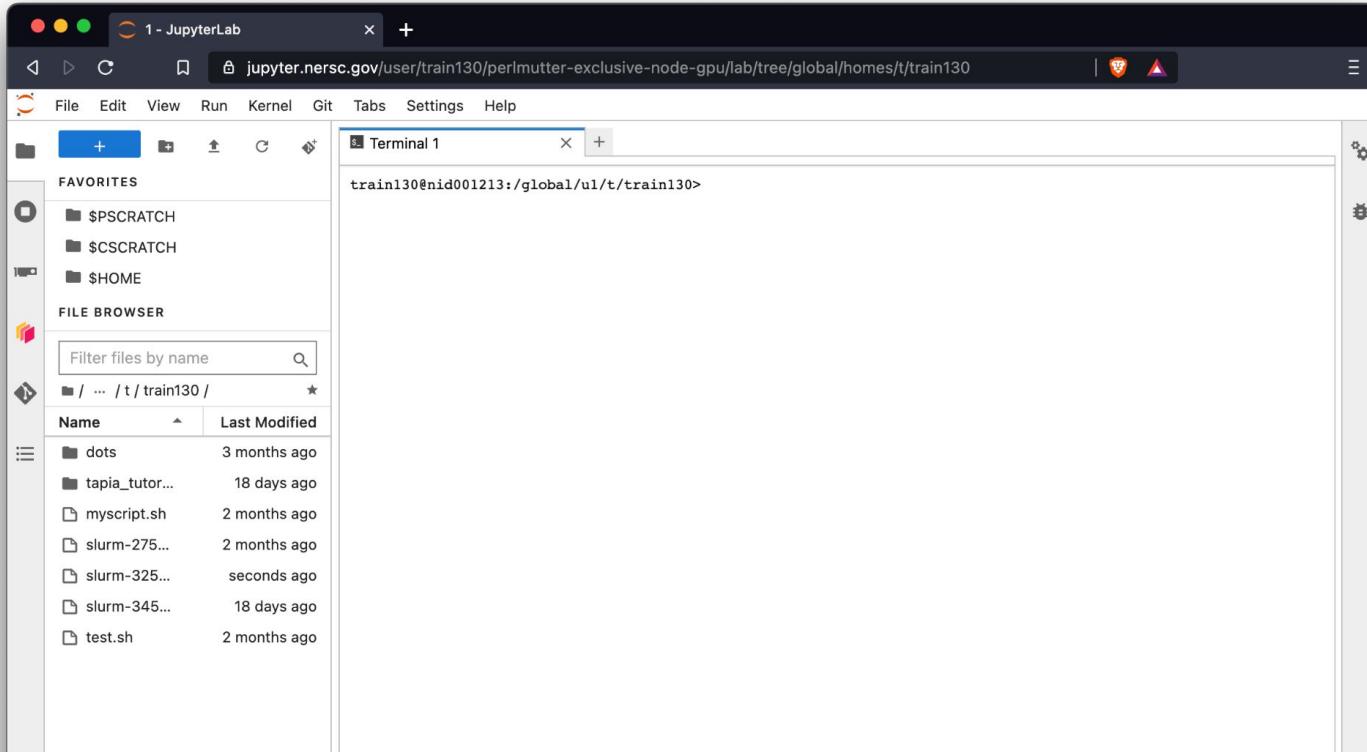
Setup

- Jupyter should take a minute to start:



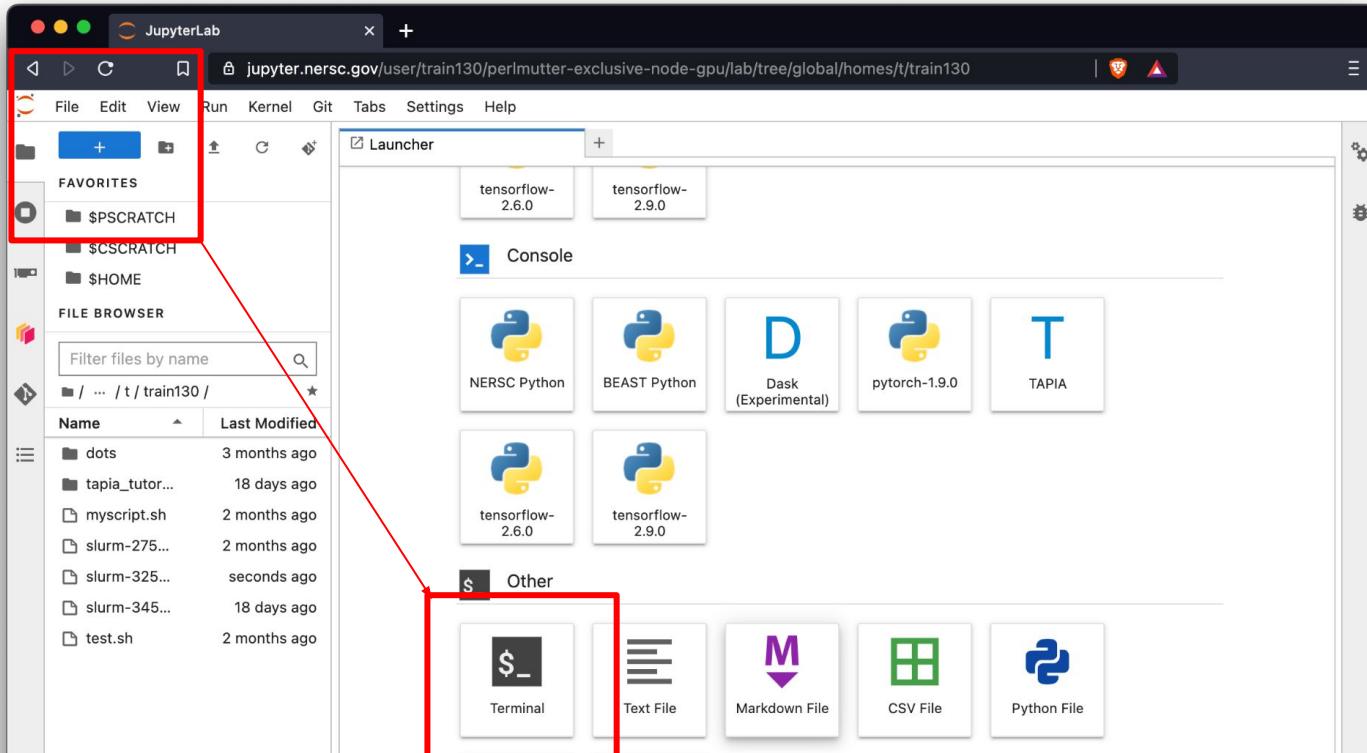
Setup

- Once started you should see a terminal



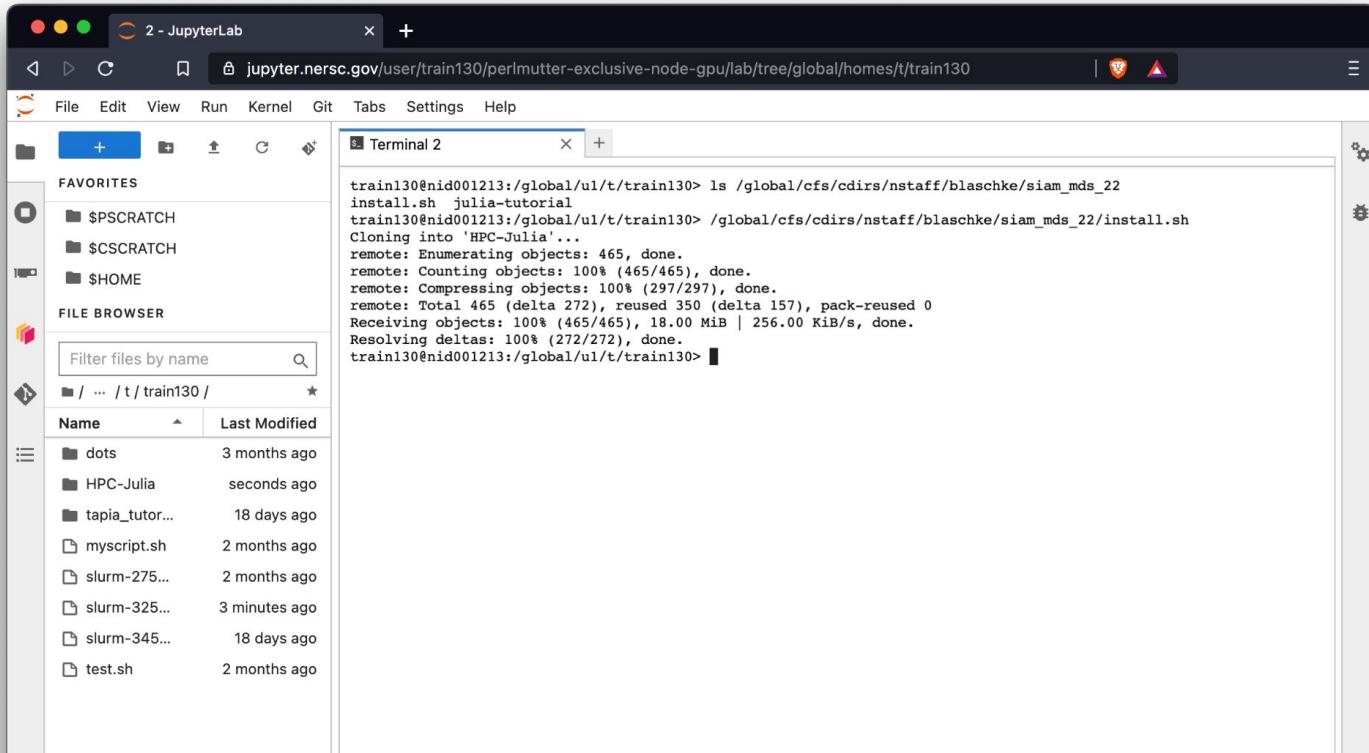
Setup

- (If you don't see a terminal), select “+” followed by “Terminal”



Setup

- (In the terminal), follow Carsten's installation instructions

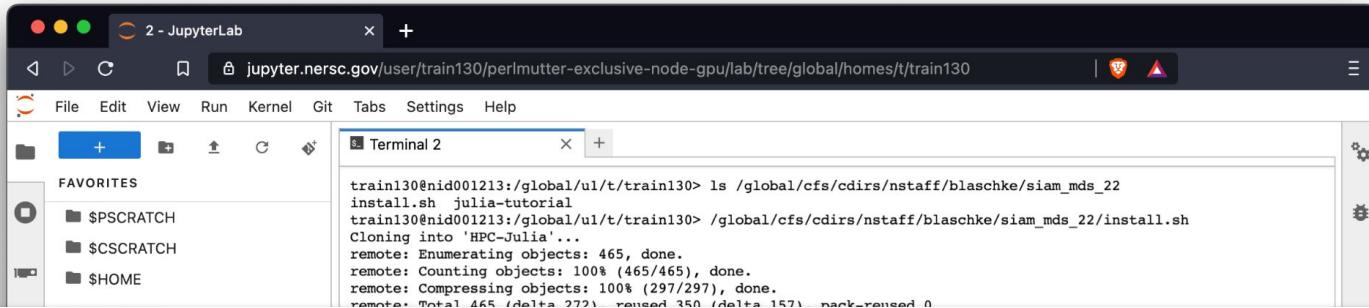


The screenshot shows the JupyterLab interface on a Mac OS X system. The title bar reads "2 - JupyterLab". The left sidebar has "FAVORITES" with "\$PSCRATCH", "\$CSCRATCH", and "\$HOME" entries. The "FILE BROWSER" section shows a directory tree under "/t/train130/" with files like "dots", "HPC-Julia", "tapia_tutor...", "myscript.sh", "slurm-275...", "slurm-325...", "slurm-345...", and "test.sh". The main area is titled "Terminal 2" and contains the following terminal session:

```
train130@nid001213:/global/u1/t/train130> ls /global/cfs/cdirs/nstaff/blaschke/siam_mds_22
install.sh julia-tutorial
train130@nid001213:/global/u1/t/train130> /global/cfs/cdirs/nstaff/blaschke/siam_mds_22/install.sh
Cloning into 'HPC-Julia'...
remote: Enumerating objects: 465, done.
remote: Counting objects: 100% (465/465), done.
remote: Compressing objects: 100% (297/297), done.
remote: Total 465 (delta 272), reused 350 (delta 157), pack-reused 0
Receiving objects: 100% (465/465), 18.00 MiB | 256.00 KiB/s, done.
Resolving deltas: 100% (272/272), done.
train130@nid001213:/global/u1/t/train130>
```

Setup

- (In the terminal), follow William's installation instructions



A screenshot of a Mac OS X desktop showing a JupyterLab interface. The title bar says "2 - JupyterLab". The address bar shows the URL "jupyter.nersc.gov/user/train130/perlmutter-exclusive-node-gpu/lab/tree/global/homes/t/train130". The left sidebar has a "FAVORITES" section with icons for "\$PSCRATCH", "\$CSCRATCH", and "\$HOME". The main area contains a terminal window titled "Terminal 2" with the following command and output:

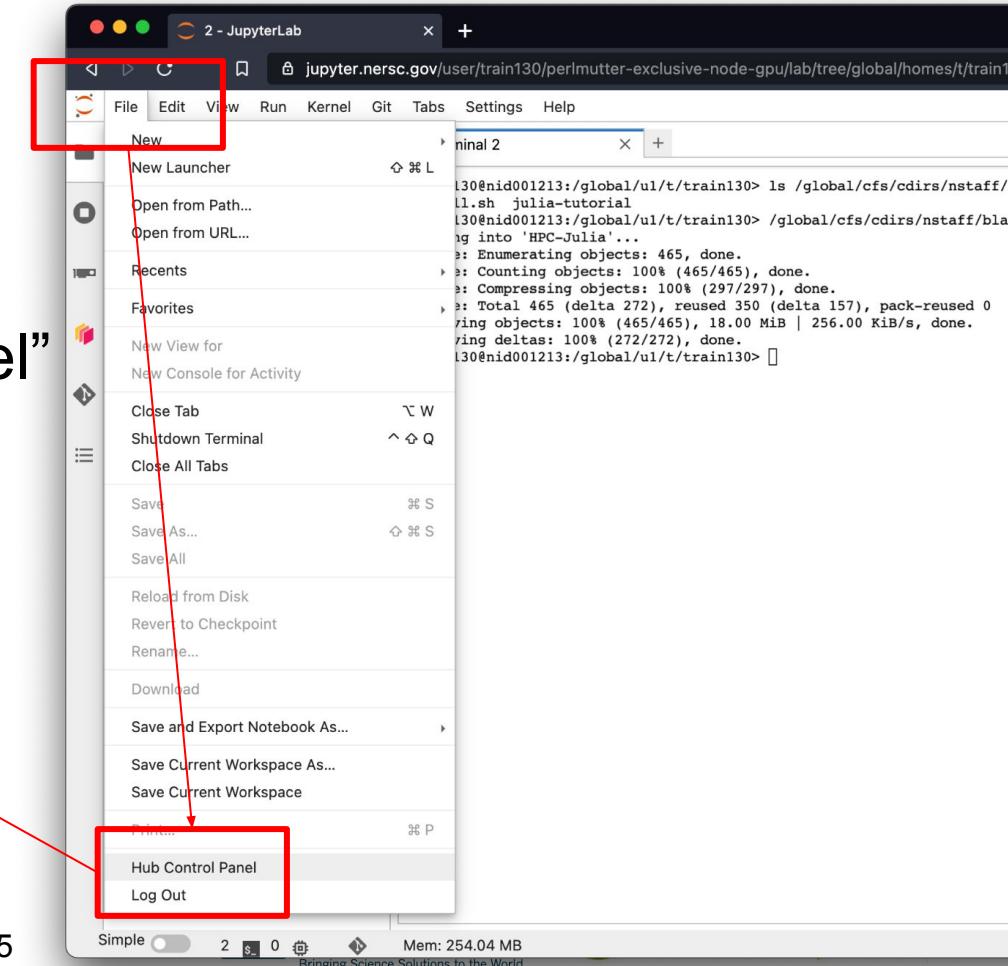
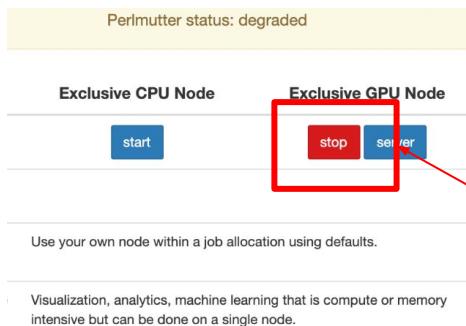
```
train130@nid001213:/global/u1/t/train130> ls /global/cfs/cdirs/nstaff/blaschke/siam_mds_22
install.sh julia-tutorial
train130@nid001213:/global/u1/t/train130> /global/cfs/cdirs/nstaff/blaschke/siam_mds_22/install.sh
Cloning into 'HPC-Julia'...
remote: Enumerating objects: 465, done.
remote: Counting objects: 100% (465/465), done.
remote: Compressing objects: 100% (297/297), done.
remote: Total 465 (delta 272), reused 350 (delta 157), pack-reduced 0
```

NERSC Julia Kernel install script here:

```
$ git clone https://github.com/JuliaHPC/juliacon24-hpcworkshop.git
$ cd juliacon24-hpcworkshop.git
$ install.sh
```

Setup

- Stop your server by going to:
“File” > “Hub Control Panel”
- Then selecting “Stop”



Setup

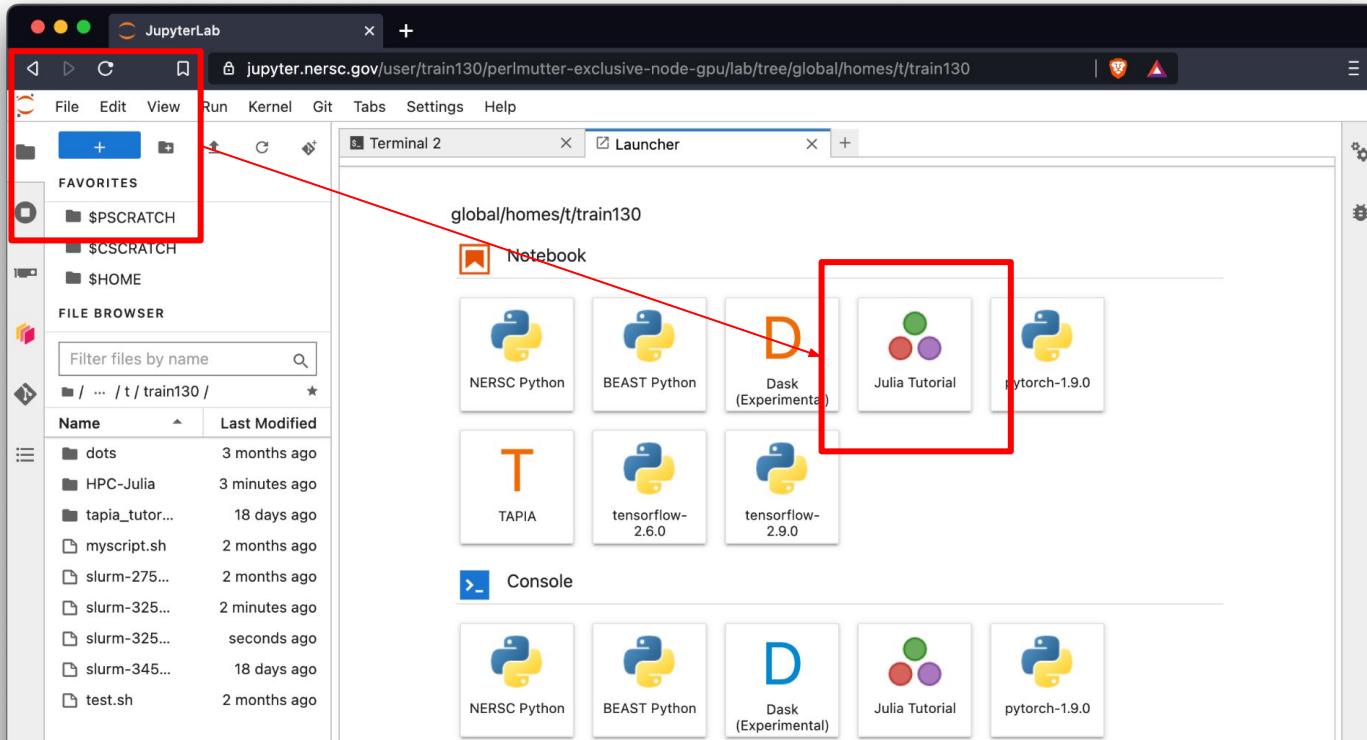
- Start a fresh exclusive node on Perlmutter

The following NERSC resources that Jupyter depends on appear to be in maintenance or having issues. This may impact Jupyter. See the [NERSC MOTD](#) for further information.
Perlmutter status: degraded

	Shared CPU Node	Exclusive CPU Node	Exclusive GPU Node	Configurable GPU
Perlmutter	start	start	start	start
Cori	start			start
<i>Resources</i>	Use a node shared with other users' notebooks but outside the batch queues.	Use your own node within a job allocation using defaults.	Use multiple compute nodes with specialized settings.	
<i>Use Cases</i>	Visualization and analytics that are not memory intensive and can run on just a few cores.	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.	

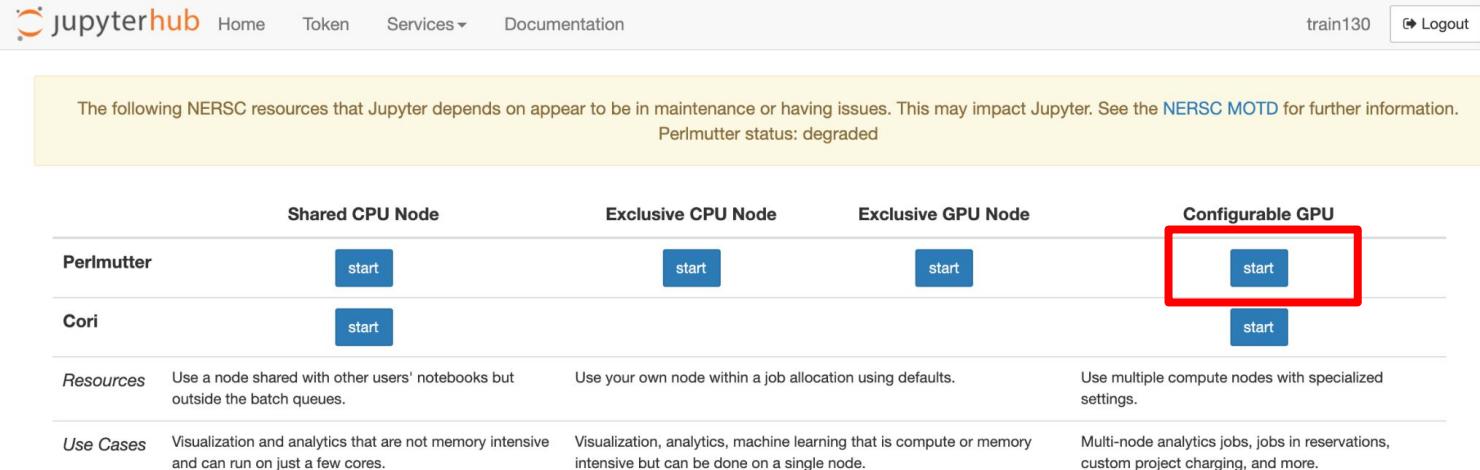
Setup

- Select the launcher, you should see the Julia Kernel :)



Using Reservations in Tutorials

- Go to <https://jupyter.nersc.gov> and select: “Configurable GPU” in the “Perlmutter” row



The following NERSC resources that Jupyter depends on appear to be in maintenance or having issues. This may impact Jupyter. See the [NERSC MOTD](#) for further information.
Perlmutter status: degraded

	Shared CPU Node	Exclusive CPU Node	Exclusive GPU Node	Configurable GPU
Perlmutter	start	start	start	start
Cori	start			start
Resources	Use a node shared with other users' notebooks but outside the batch queues.	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.	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.

Jupyter Options:

Leave defaults, except:

Account:

ntrain1

Reservation

julia_traing

Time

180

The screenshot shows a web browser window titled "JupyterHub" with the URL "jupyter.nersc.gov/hub/spawn/train130/perlmutter-configurable-gpu". The page displays a message about NERSC resources being in maintenance or having issues, specifically mentioning "Perlmutter status: degraded". Below this, the title "Server Options" is centered. The configuration form includes the following fields:

- Account ("_g" suffix will be added as needed): ntrain8
- Constraint: gpu
- QOS: jupyter
- cpus-per-task (node has 128 cpus): 128
- gpus-per-task (node has 4 GPUs): 4
- nodes (maximum of 4 for jupyter QOS): 1
- ntasks-per-node: 1
- Reservation: siam_intro_gnn
- time (time limit in minutes): 120

A large orange "Start" button is located at the bottom of the form.



Note:

- Reservations now run for 3 hours. If you request a node during this time, make sure that the current time + requested time does not exceed the reservation's end time.
- Training accounts are fully fledged NERSC accounts
- Will remain available until Sunday – please copy any data and code to your computer before then

