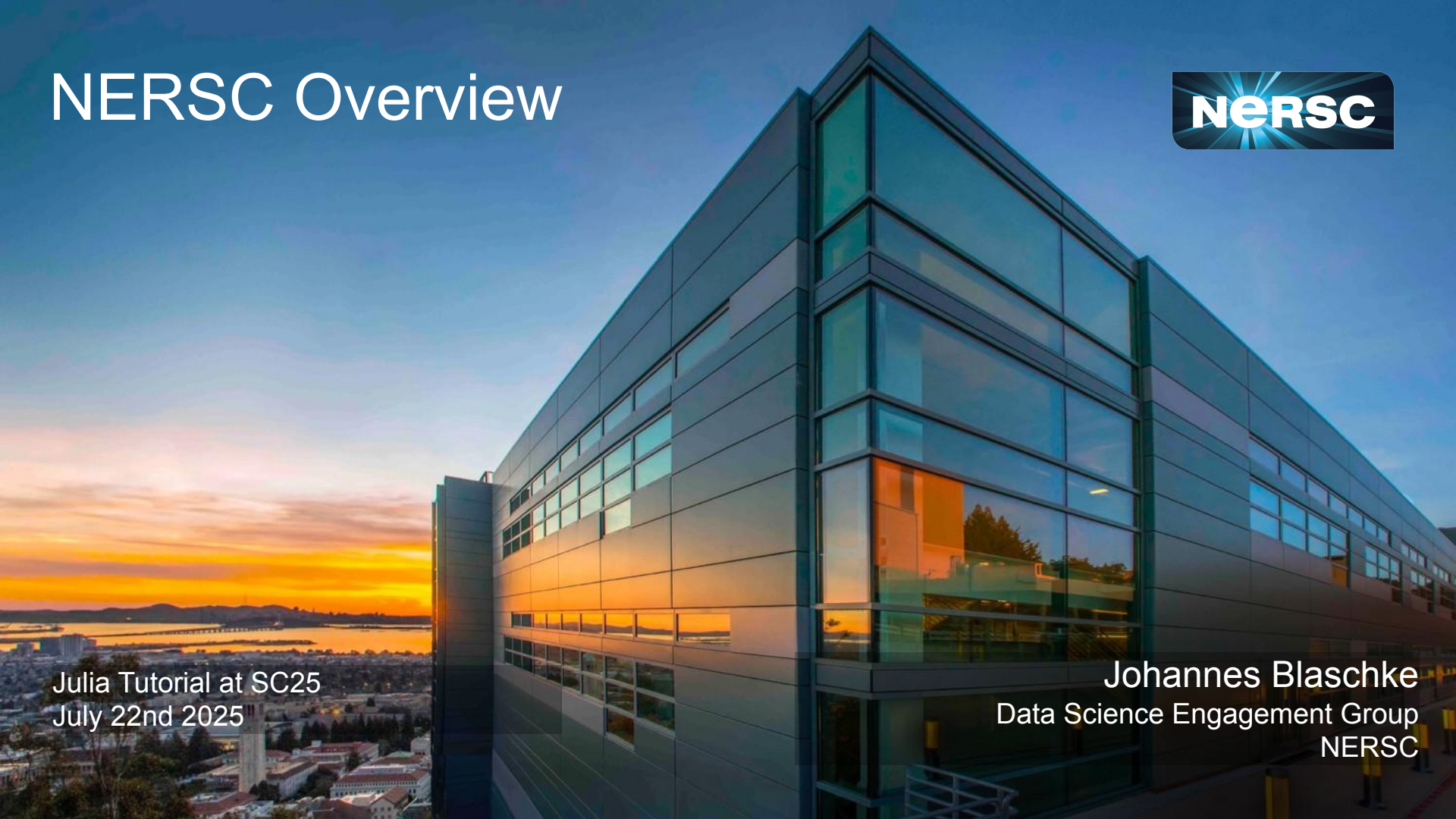
NERSC Overview



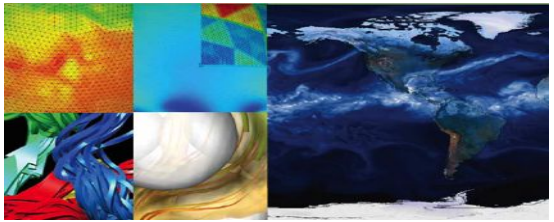
Julia Tutorial at SC25
July 22nd 2025

Johannes Blaschke
Data Science Engagement Group
NERSC

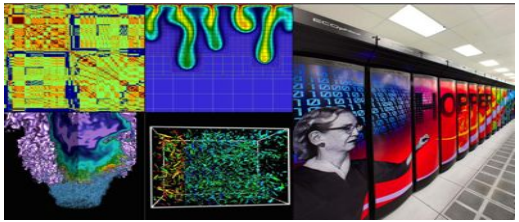
NERSC: Mission HPC for DOE Office of Science Research

U.S. DEPARTMENT OF ENERGY | Office of Science

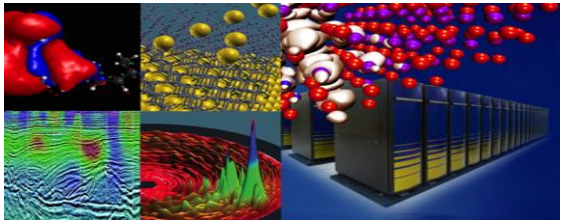
Largest funder of physical science research in the U.S.



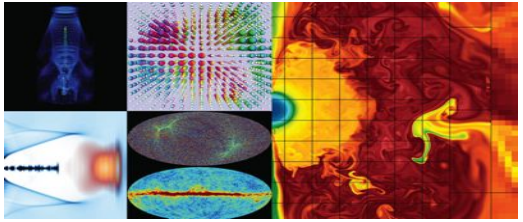
Biological and Environmental Research



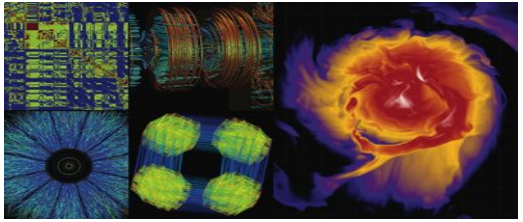
Computing



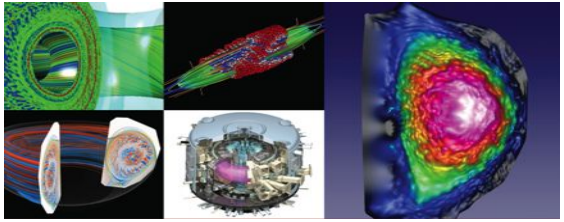
Basic Energy Sciences



High Energy Physics



Nuclear Physics

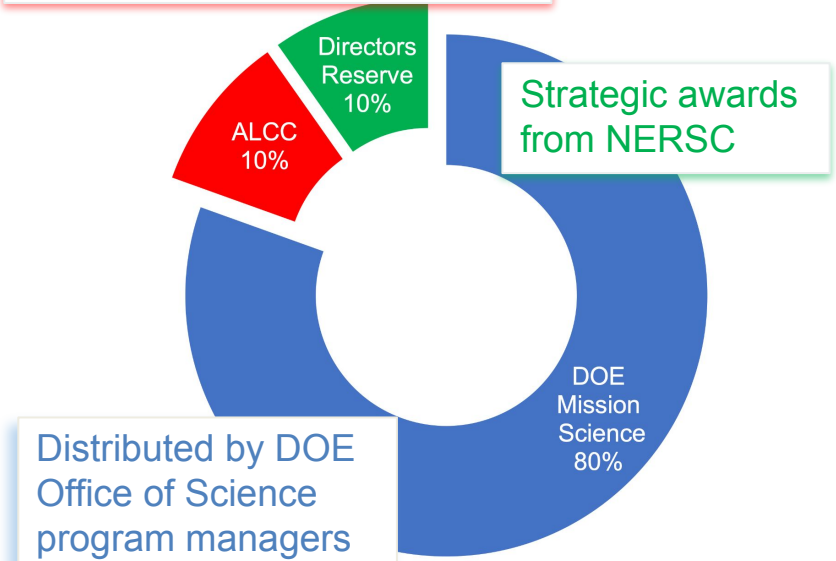


Fusion Energy, Plasma Physics

NERSC Directly Supports Office of Science Priorities

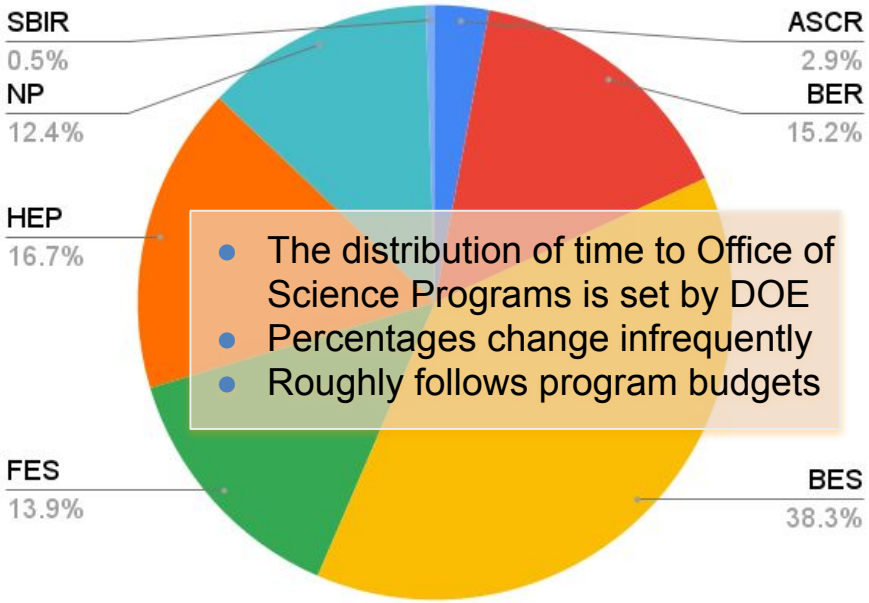
2023 Allocation Breakdown (Hours Millions)

Competitive awards run by DOE Advanced Scientific Computing Research Office



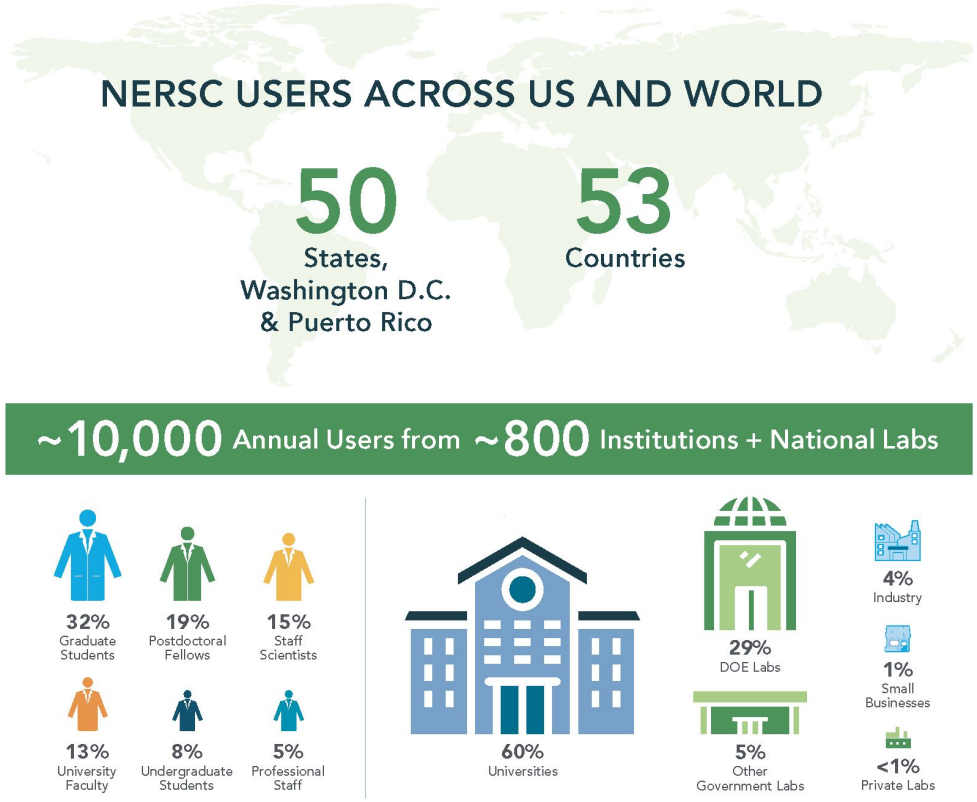
Strategic awards from NERSC

Distributed by DOE Office of Science program managers



- The distribution of time to Office of Science Programs is set by DOE
- Percentages change infrequently
- Roughly follows program budgets

NERSC Turns 50 This Year!

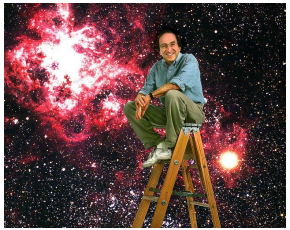


CDC 6600 at LLNL 1974

Success Is Depth and Breadth of Scientific Impact

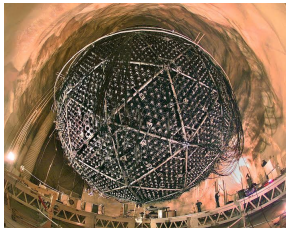
An Accelerating Universe
Saul Perlmutter, Berkeley Lab

Perlmutter's Nobel winning team is believed to have been the first to use supercomputers to analyze and validate observational data in cosmology, contributing to the discovery of the accelerating expansion of the universe.



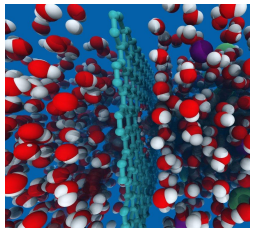
Oscillating Neutrinos
Sudbury Neutrino Observatory (SNO)

Data from SNO was transferred to NERSC and analyzed in what became known as the "West Coast Analysis" leading to the discovery of neutrino oscillations and a Nobel Prize.



New Approach to Water Desalinization
Jeff Grossman, MIT

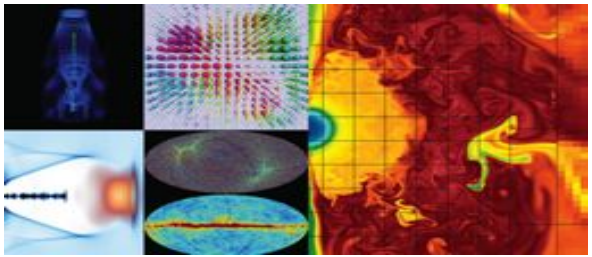
One of Smithsonian Magazine's Top 5 Most Surprising Milestones of 2012 was the computationally driven discovery of an approach to desalination of water that is more efficient and less expensive than existing systems.



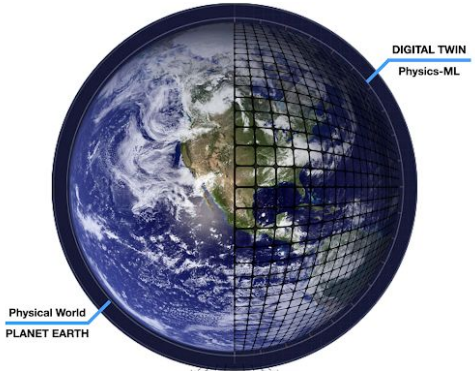
NERSC has been acknowledged in 8,790 refereed scientific publications & high profile journals since 2020

- Nature [47]
- Nature Family of Journals [463]
- Proc. of the National Academy of Sciences [222]
- Science [36]
- Monthly Notices of the Royal Astron. Society [397]
- Physical Review* [4,170]
- Astrophysical Journal [792]
- Physics of Plasmas [685]

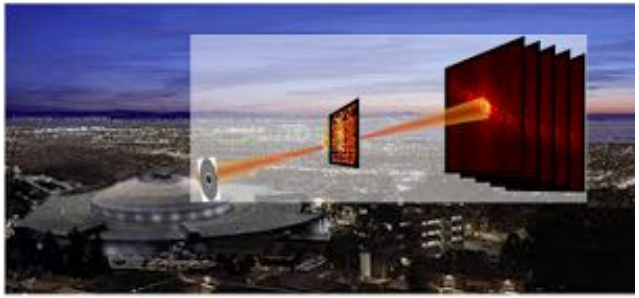
We Accelerate Scientific Discovery for Thousands of Office of Science Users with 3 Advanced Capability Thrusts



Large-scale applications for simulation, modeling and data analysis



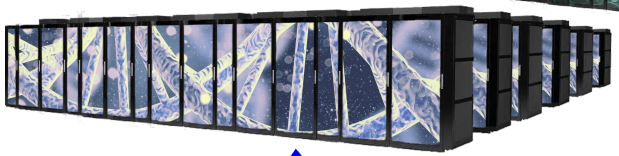
Complex experimental and AI driven workflows



Time-sensitive and interactive computing

The NERSC workload is diverse with growing emphasis on integrated research workflows

Responding to the DOE Mission



NERSC-8: Cori
Manycore CPU architectures

2016

NERSC-9: Perlmutter
CPU and GPU nodes
Expanded Simulation, Learning & Data

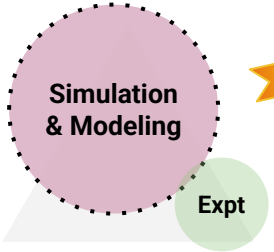
2020

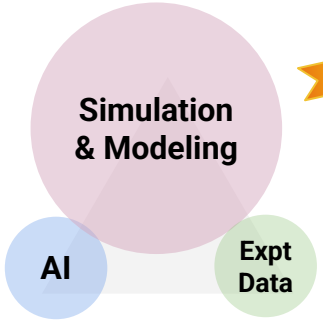
NERSC-10:
Accelerating end-to-end workflows

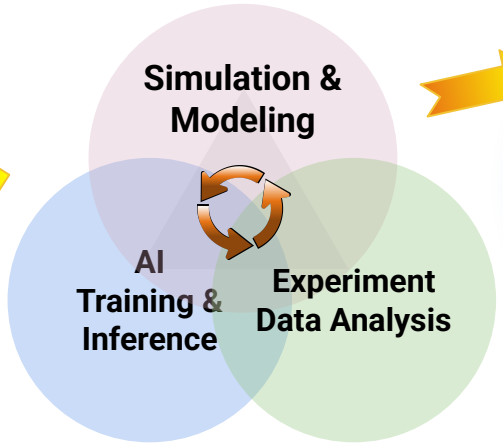
2026

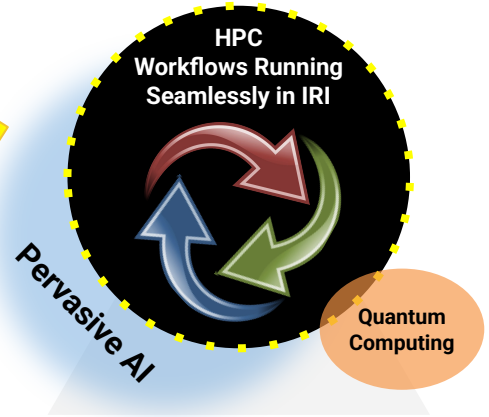
NERSC-11:
Beyond Moore

2030+

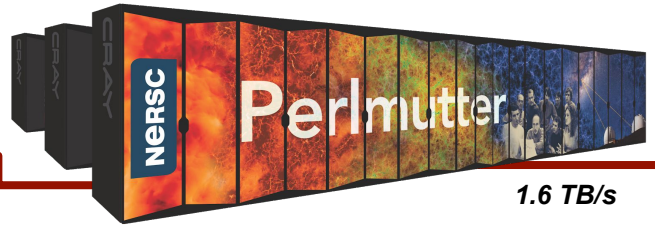








NERSC Systems Ecosystem



#25 (#6), 93.8PF Peak

>5 *TB/s*

35 PB All-Flash Scratch

HPE Slingshot 11 interconnect
4 NICs/GPU node,
1 NIC/CPU node

1,792 GPU-accelerated nodes
4 NVIDIA A100 GPUs+1 AMD "Milan" CPU
448 TB (CPU) + 320 TB (GPU) memory

3,072 CPU-only nodes
2 AMD "Milan" CPUs
1,536 TB CPU memory

1.6 TB/s

Off-Platform Storage

50 GB/s — HPSS Tape Archive ~300 PB

100 GB/s — Common File System 130 PB

5 GB/s — /home 450 TB

DTNs, Gateways

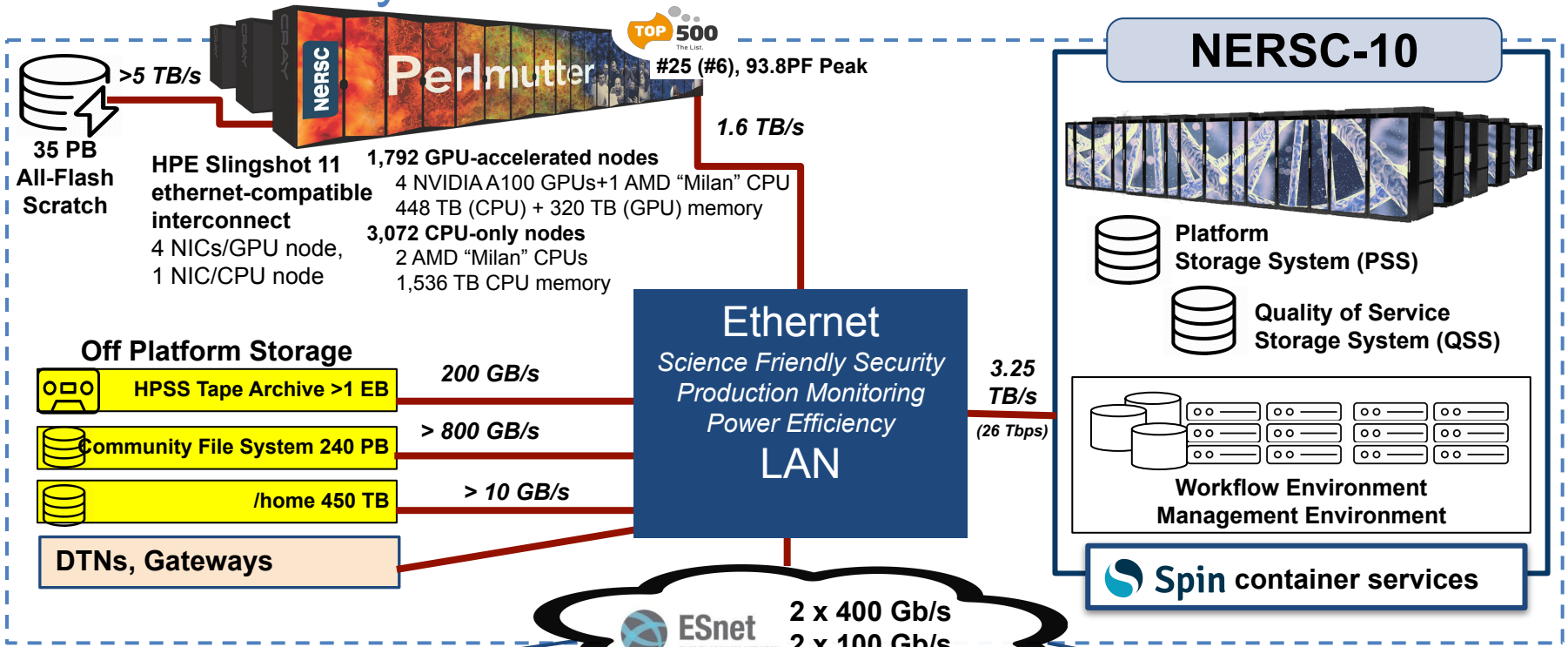
Spin edge services

Ethernet
Science Friendly Security
Production Monitoring
Power Efficiency
LAN

ESnet ENERGY SCIENCES NETWORK
2 x 400 Gb/s
2 x 100 Gb/s

Experimental Facility | ASCR Facility | Home Institution | Cloud | Edge

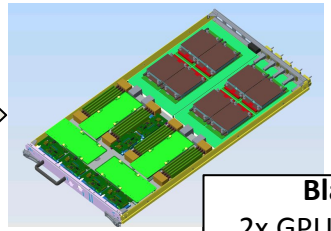
NERSC Ecosystem in 2027

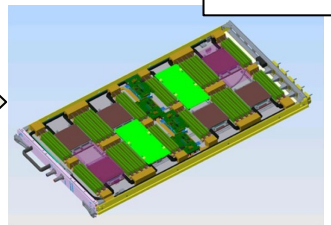


Perlmutter system configuration

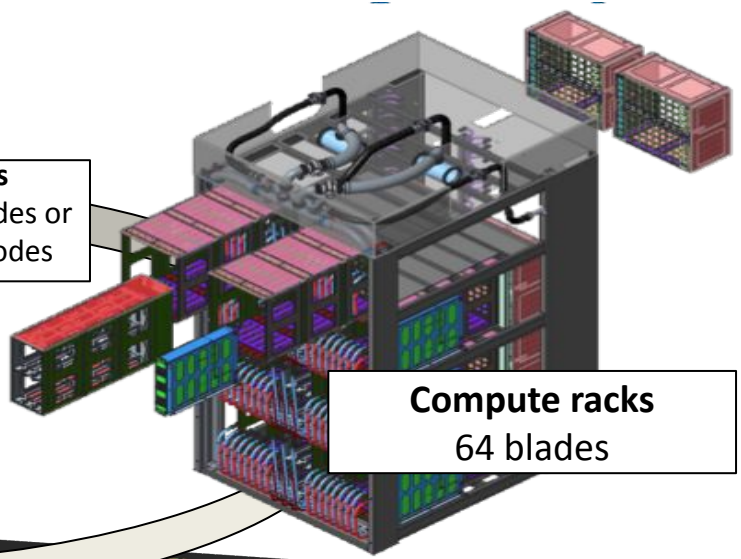
NVIDIA "Ampere" GPU Nodes
4x GPU + 1x CPU
40 GiB HBM + 256 GiB DDR
4x 200G "Slingshot" NICs

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





Blades
2x GPU nodes or
4x CPU nodes

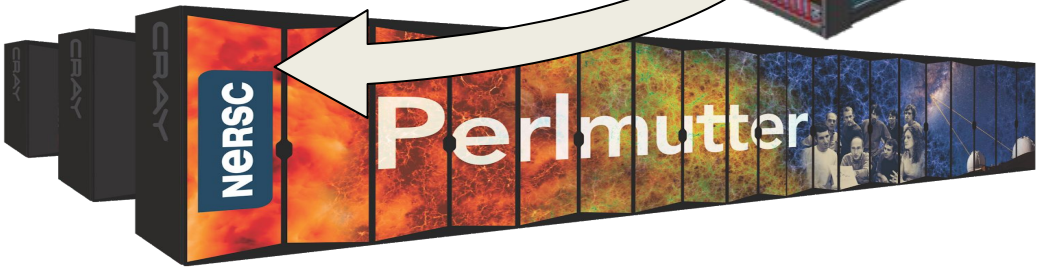


Compute racks
64 blades

Centers of Excellence
Network
Storage
App. Readiness
System SW

→

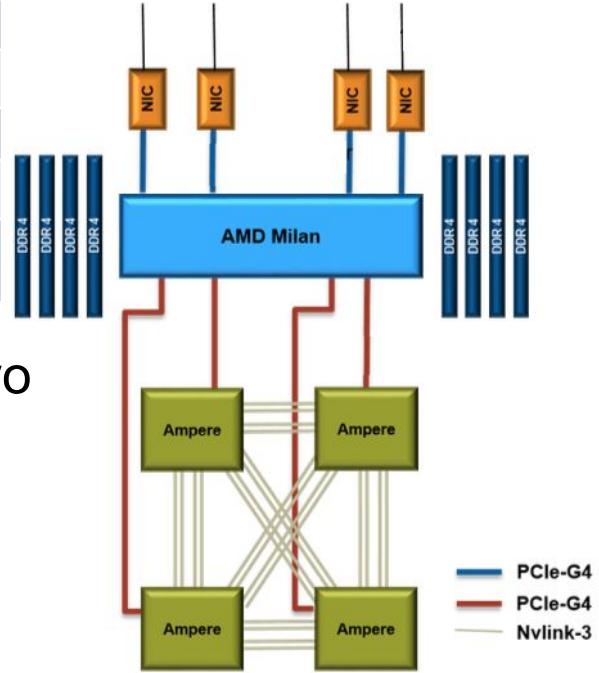
Perlmutter system
GPU racks
CPU racks
~6 MW



Perlmutter Node Configuration

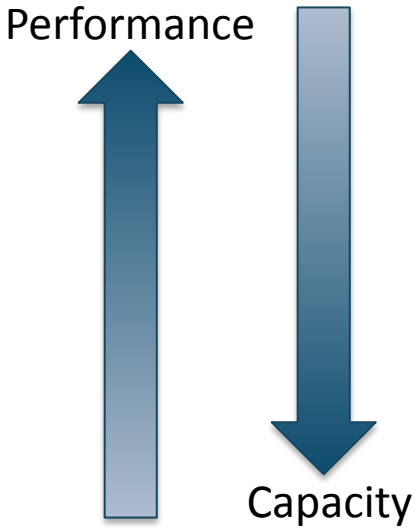
Partition	Nodes	CPU	RAM	GPU	NIC
GPU	1536	1x AMD EPYC 7763	256GB	4x NVIDIA A100 (40GB)	4x HPE Slingshot 11
	256	1x AMD EPYC 7763	256GB	4x NVIDIA A100 (80GB)	4x HPE Slingshot 11
CPU	3072	2x AMD EPYC 7763	512GB	–	1x HPE Slingshot 11
Login	40	1x AMD EPYC 7713	512GB	4x NVIDIA A100 (40GB)	–
Large Memory	4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

GPU Node Architecture:



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

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
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 (8 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

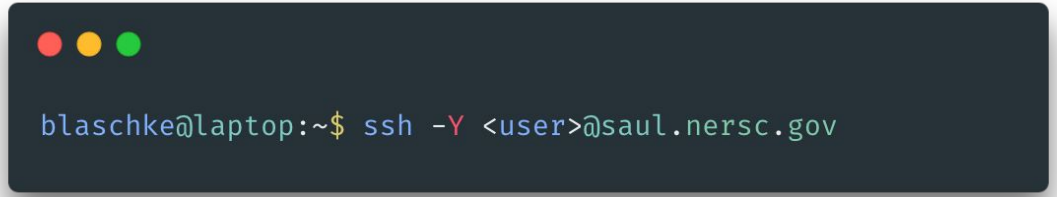
There are many different ways to access NERSC. To use our resources, you need to either:

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” (see: https://docs.nersc.gov/services/sfapi/)

Connecting with SSH

- To access Perlmutter, use:
  `saul.nersc.gov`
  `perlmutter.nersc.gov`

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



Connecting with SSH

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

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

From here you can interact with perlmutter…

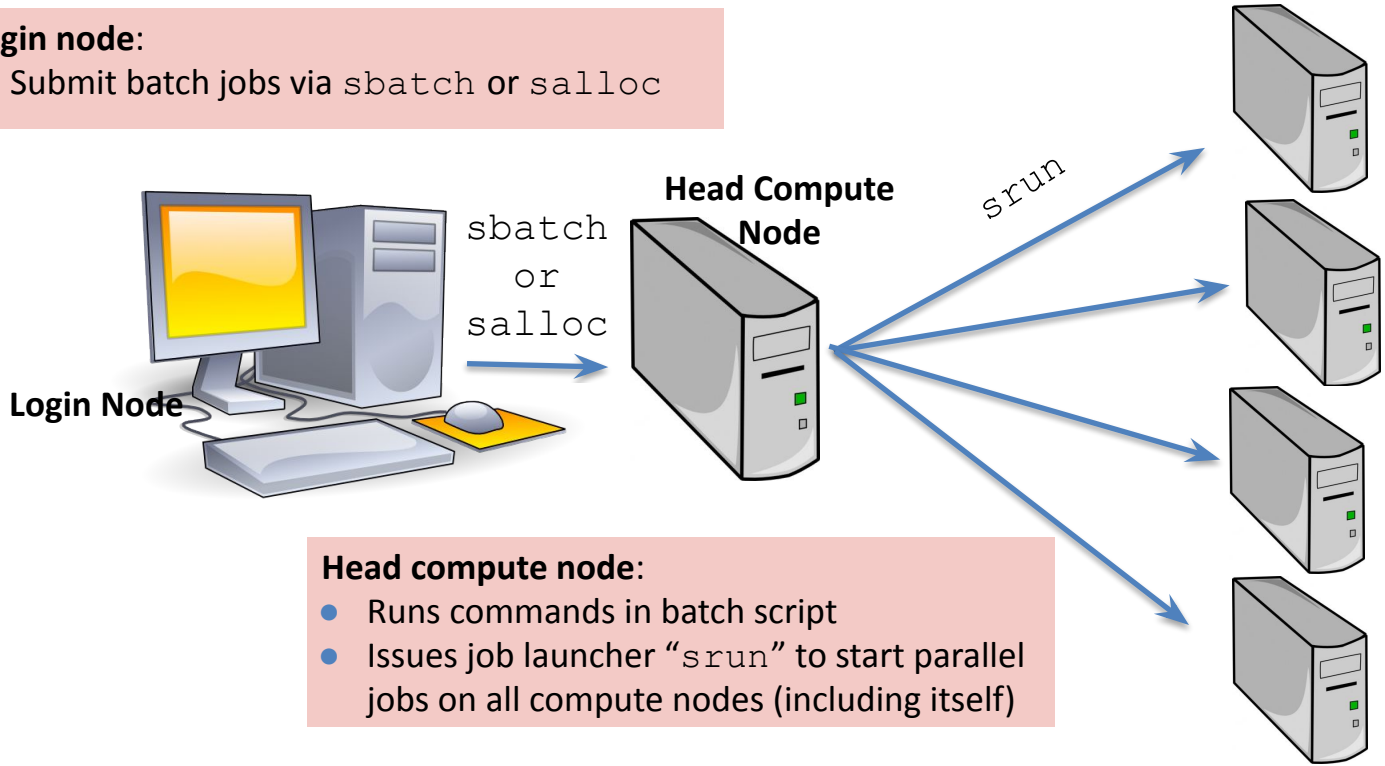
```
blaschke@laptop:~$ ssh blaschke@saul.nersc.gov
********************************************************************************
                        NOTICE TO USERS

Lawrence Berkeley National Laboratory operates this computer system under
contract to the U.S. Department of Energy.  This computer system is the
property of the United States Government and is for authorized use only.
Users (authorized or unauthorized) have no explicit or implicit
expectation of privacy.

Any or all uses of this system and all files on this system may be
intercepted, monitored, recorded, copied, audited, inspected, and disclosed
to authorized site, Department of Energy, and law enforcement personnel,
as well as authorized officials of other agencies, both domestic and foreign.
By using this system, the user consents to such interception, monitoring,
recording, copying, auditing, inspection, and disclosure at the discretion
of authorized site or Department of Energy personnel.

Unauthorized or improper use of this system may result in administrative
disciplinary action and civil and criminal penalties. By continuing to use
this system you indicate your awareness of and consent to these terms and
conditions of use. LOG OFF IMMEDIATELY if you do not agree to the conditions
stated in this warning.

********************************************************************************

Login connection to host x3114c0s15b0n0:


################################################################################
                                                 _   _
                                                | | | |
 _ __   ___ _ __| |_ _ __ ___  _   _| |_| |_ _ __ _ __
| '_ \ / _ \ '__| | '_ ` _ \| | | | __| __/ _ \ '__|
| |_) |  __/ |  | | | | | | | |_| | |_| ||  __/ |
| .__/ \___|_|  |_|_| |_| |_|\__,_|\__|\__\___|_|
| |
|_|

Hello, World!

Welcome to perlmutter!
################################################################################

For all planned outages, see: https://www.nersc.gov/live-status/motd/

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

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`



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

*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

run 64 processes in parallel

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

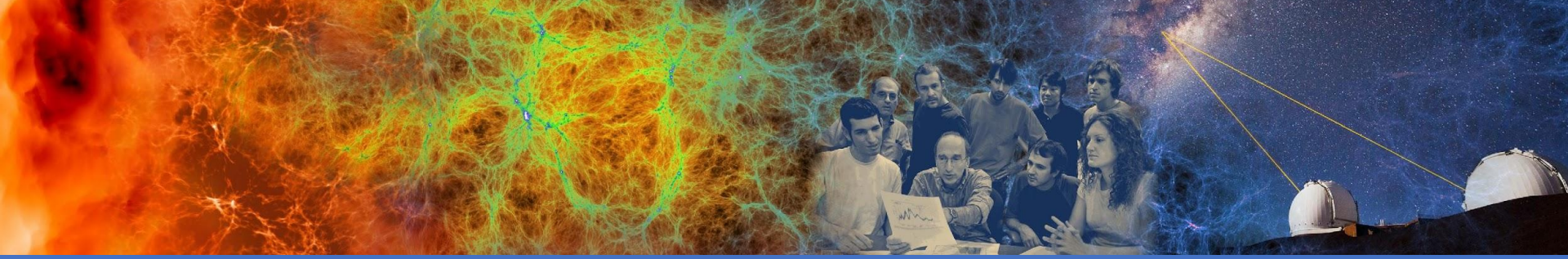
```
% sbatch my_batch_script.sh
```

To run via interactive batch

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

<wait_for_session_prompt. Land on a compute node>

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



Accessing NERSC
and submitting your first Job

Access to Perlmutter and Use Julia Module

- NERSC users have been added to `trn019` project
- Non-users were sent the instruction to get a training account
  - Account valid through July 29
- Login to Perlmutter: ssh username@perlmutter.nersc.gov
- Julia modules:
  - `% module load julia`
- Running Jobs examples:
  - https://docs.nersc.gov/jobs/

Accessing Compute Nodes

- To use 1 GPU only (sample flags for sbatch or salloc):
  - `-A trn019 -C gpu -N 1 -c 32 -G 1 -t 30:00 -q` **`shared`**
- To use multiple nodes (sample flags for sbatch or salloc):
  - `-A trn019 -C gpu -N 2 -t 30:00 -q` **`regular`**
    (or `-q` **`interactive`** for salloc)

- Or for faster queue access:
  `-q` **`premium`** At most 5 jobs in the queue (incl running)
  `-q` **`debug`** At most 5 8-node 30min jobs

Make Sure to Clone The Tutorial Repo

github.com/JuliaParallel/julia-hpc-tutorial-icpp25

Pro tip: `$HOME` is not the best file system for running jobs at scale. For large-scale jobs, add

```
export JULIA_DEPOT_PATH=$SCRATCH/depot
```

to you scripts for optimal performance (remember to do this in the config script, **and** the job scrip) … or use a container!

Make Sure to Clone The Tutorial Repo

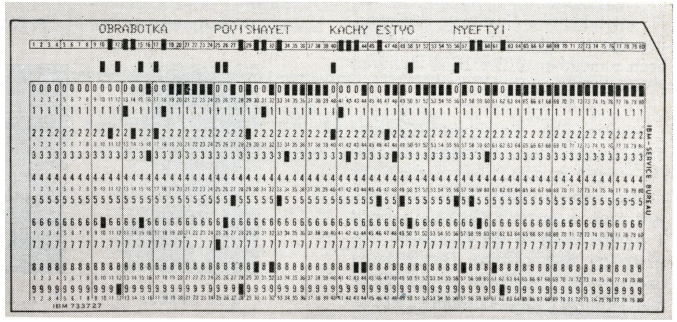
github.com/JuliaParallel/julia-hpc-tutorial-icpp25

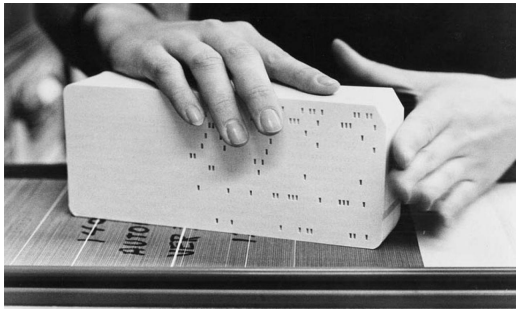
Pro tip: `$HOME` is not the best file system for running jobs at

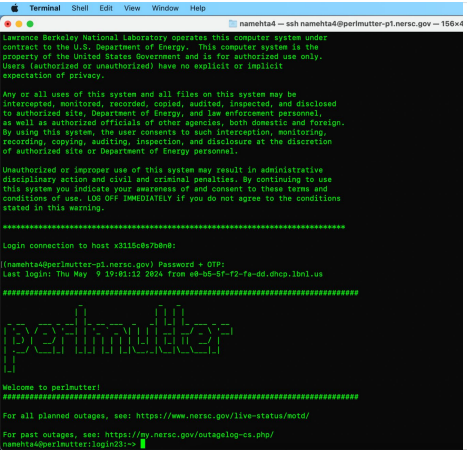
Automatically generate assets / configs by running:

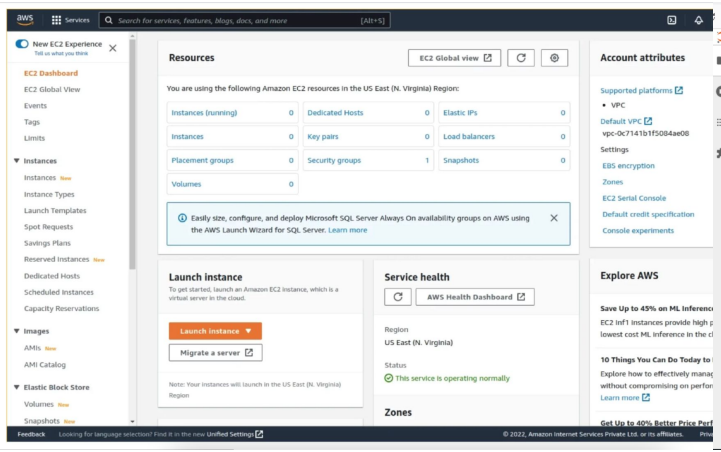
```
$ git clone git@github.com:JuliaParallel/julia-hpc-tutorial-icpp25.git
$ cd julia-hpc-tutorial-icpp25/
$ make nersc
```

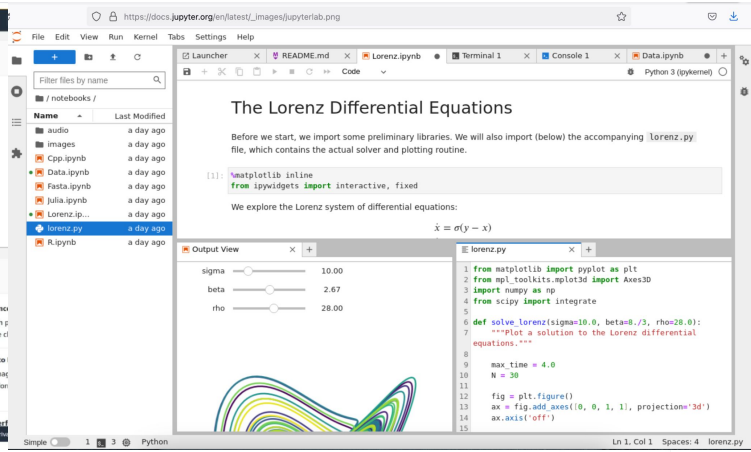
HPC User Interface





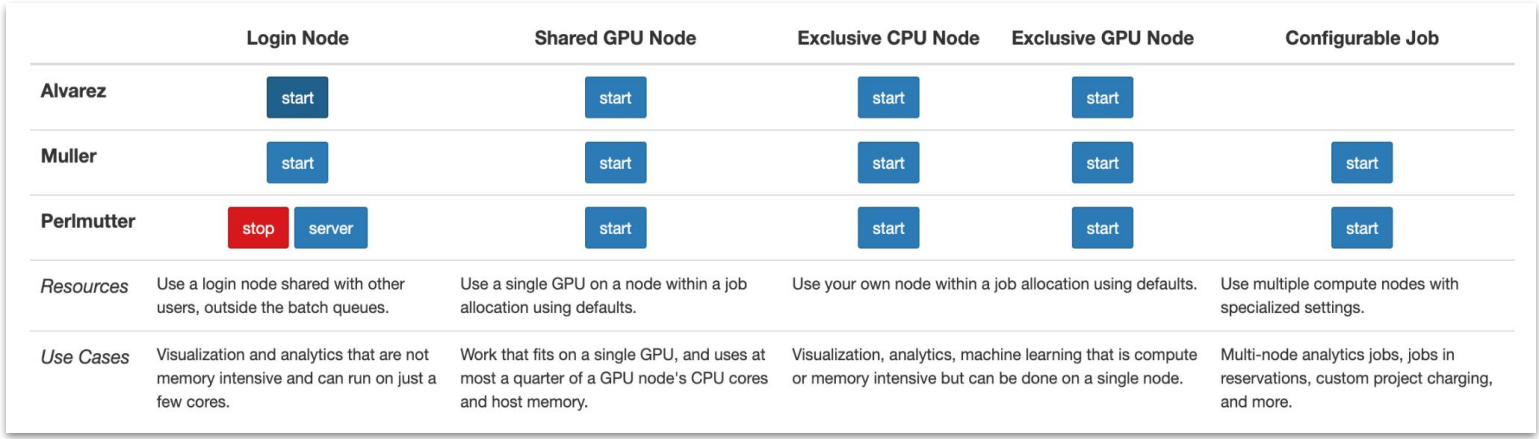






Logging into Jupyter

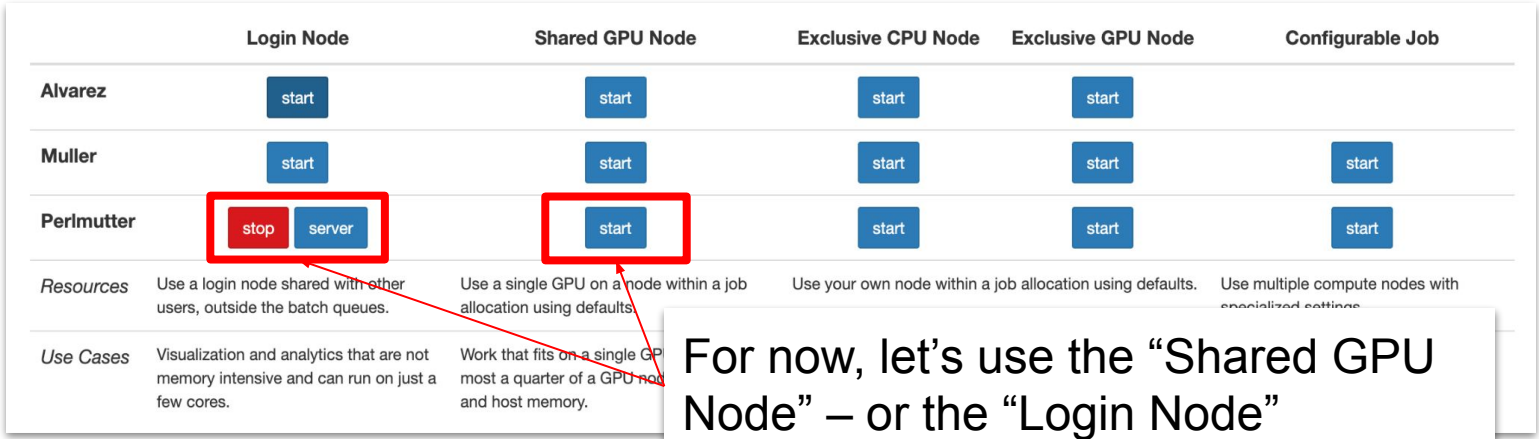
- Go to https://jupyter.nersc.gov
- Sign in using your training account credentials
- Select your preferred jupyter instance:



	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.

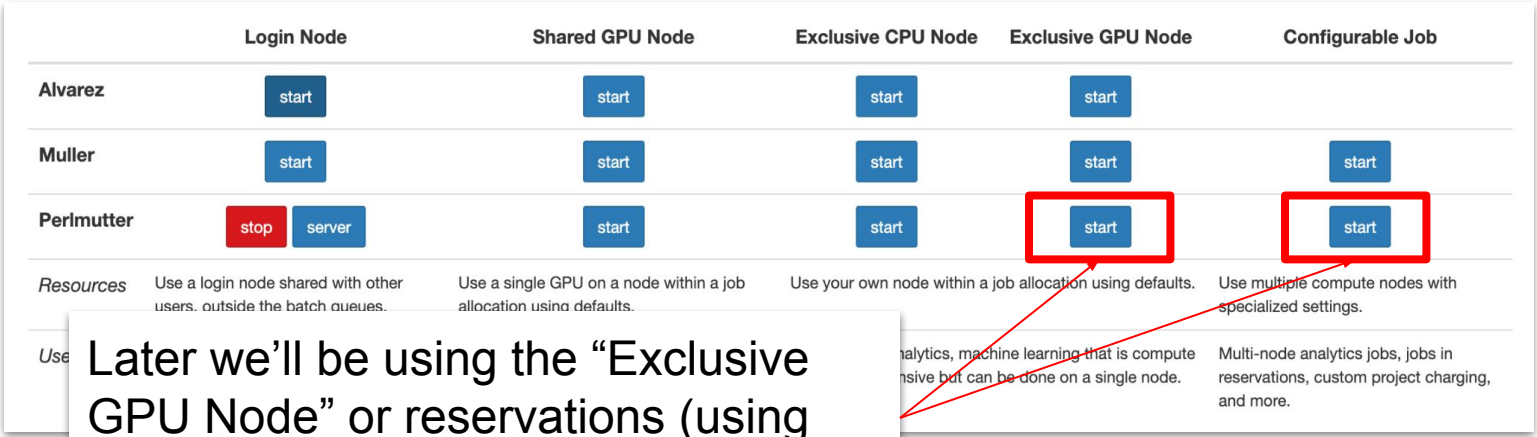
Logging into Jupyter

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



Logging into Jupyter

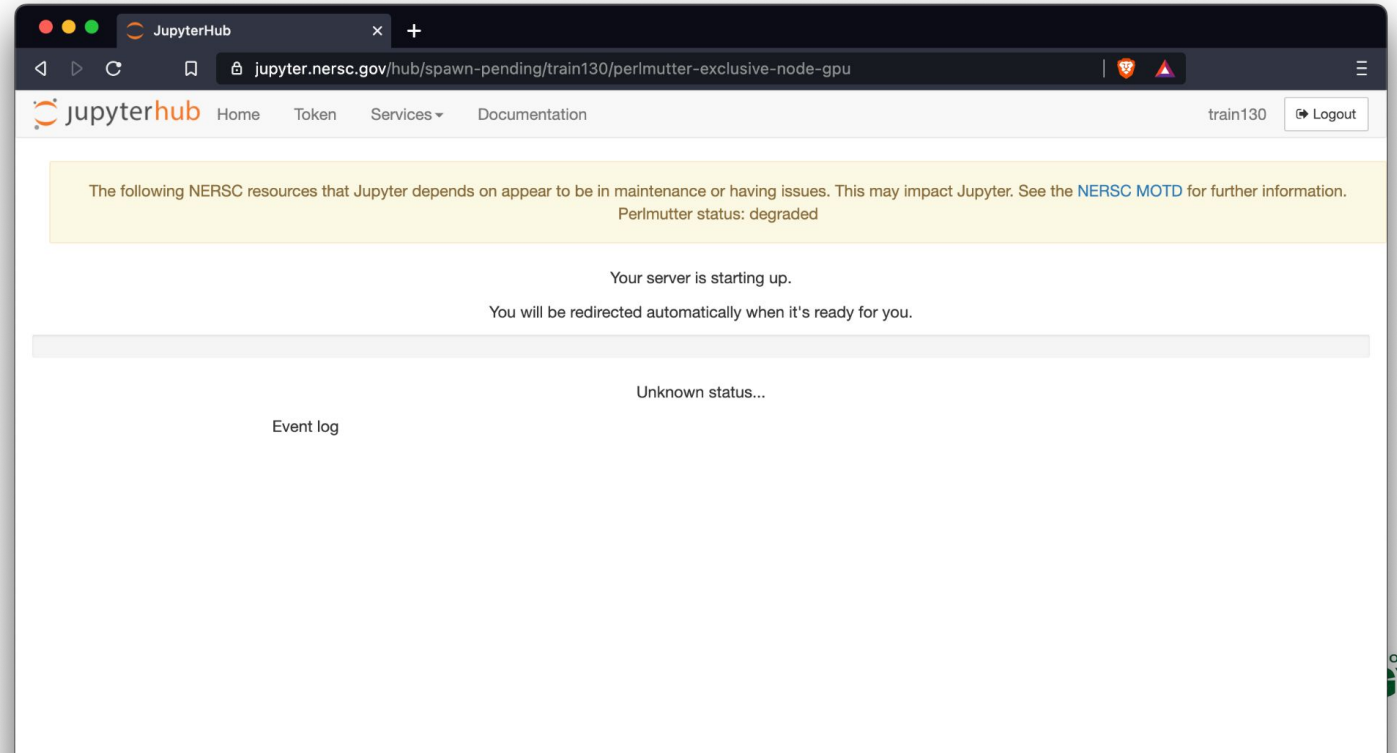
- Go to https://jupyter.nersc.gov
- Sign in using your training account credentials
- Select your preferred jupyter instance:



Later we'll be using the "Exclusive GPU Node" or reservations (using the "Configurable Job")

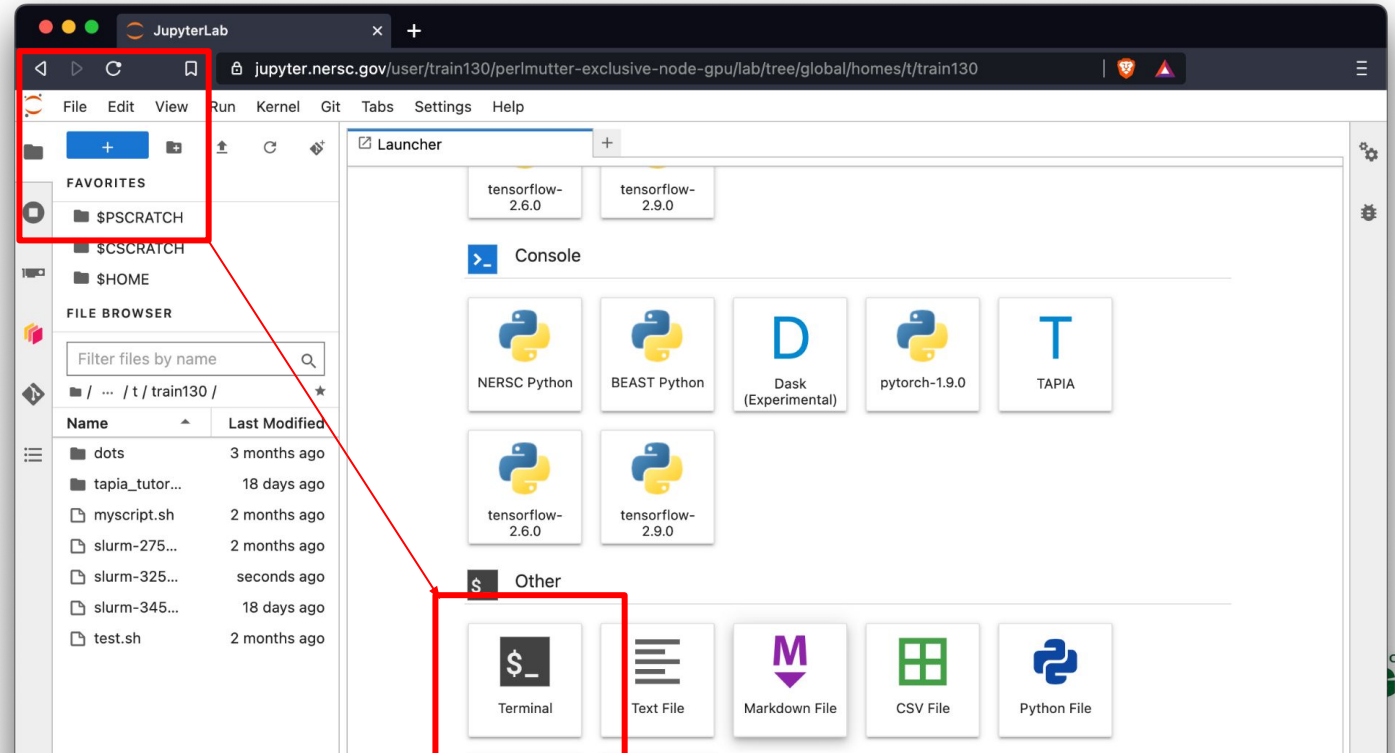
Getting a Terminal in Jupyter

- Jupyter should take a minute to start:



Getting a Terminal in Jupyter

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

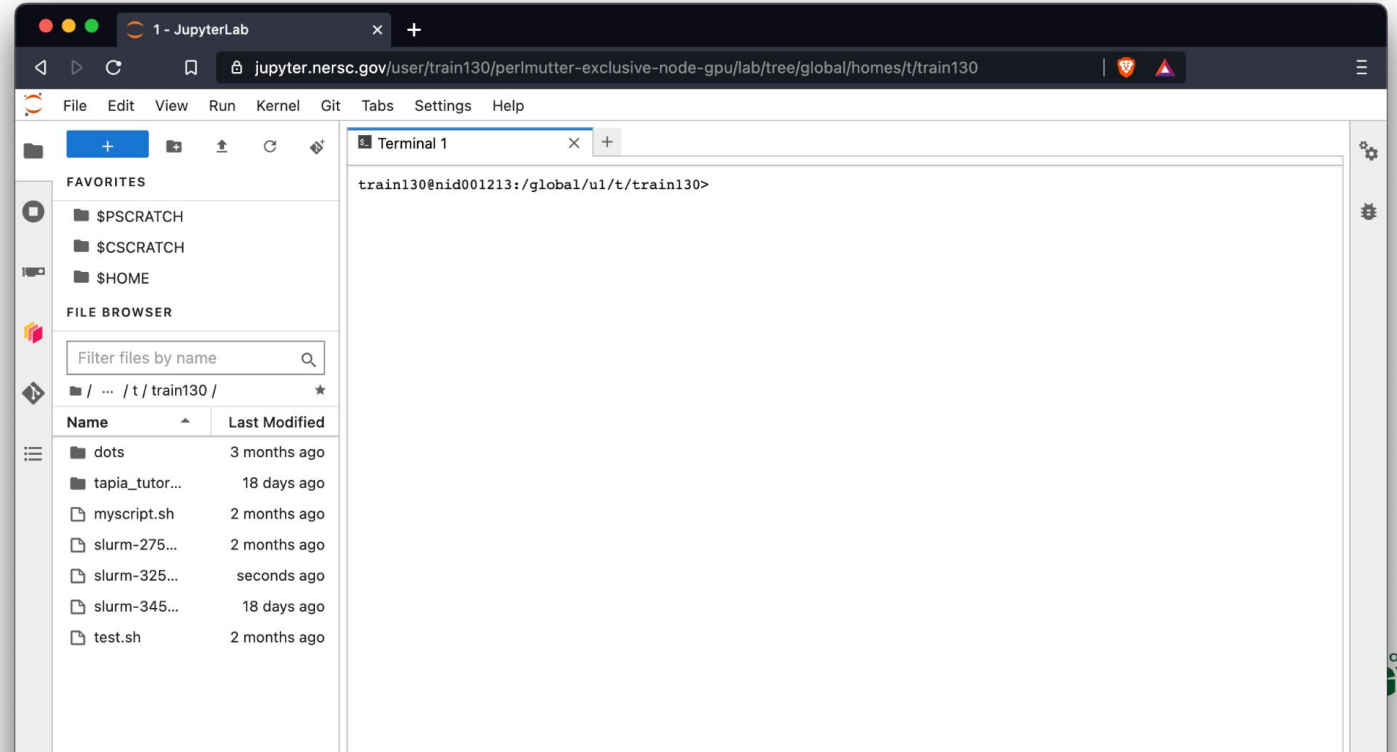


NERSC

Office of Science

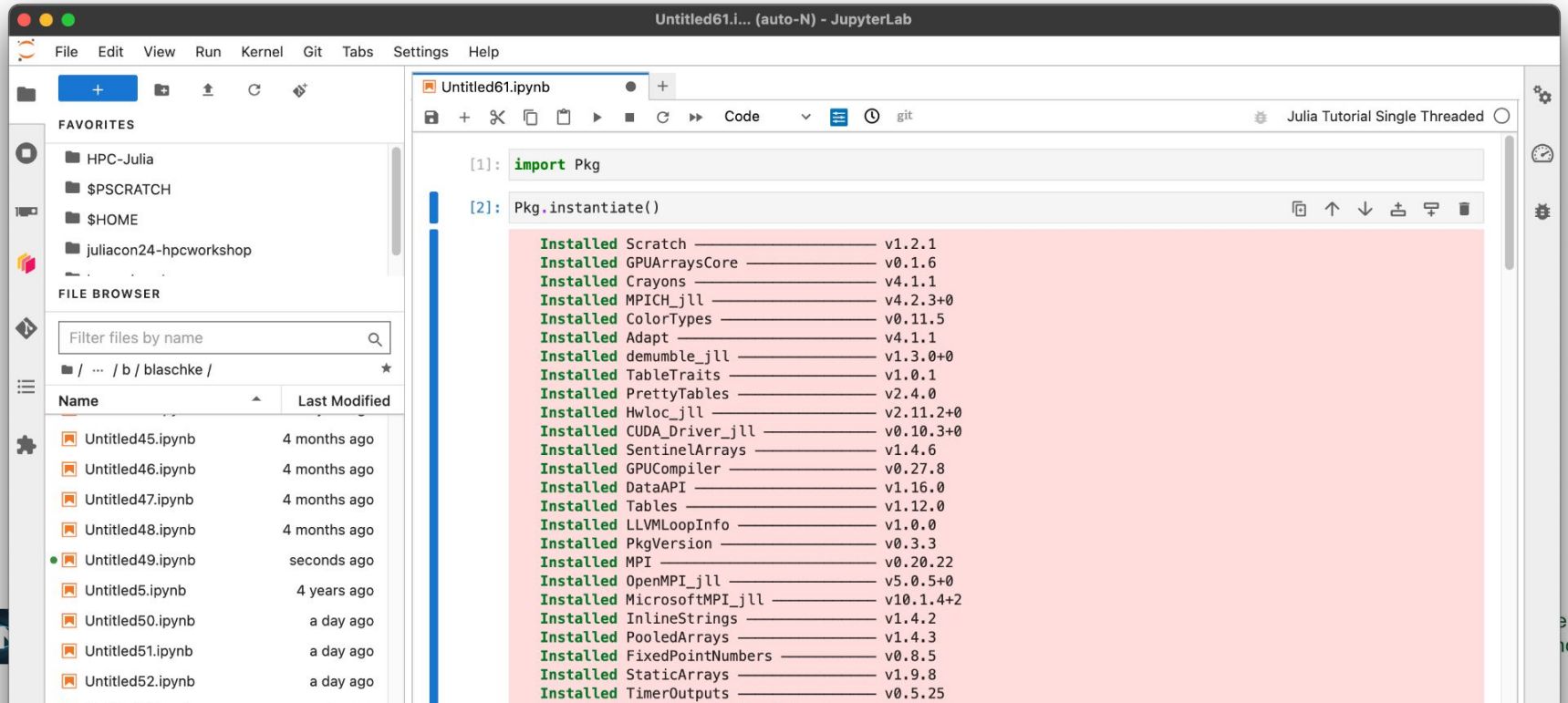
Setup

- Once started you should see a terminal



Pro Tip: Projects handle dependencies

- Run: `import Pkg; Pkg.activate("path/to/the/repo"); Pkg.instantiate()` to insure all dependencies are installed:



Pro Tip: Sanity Check

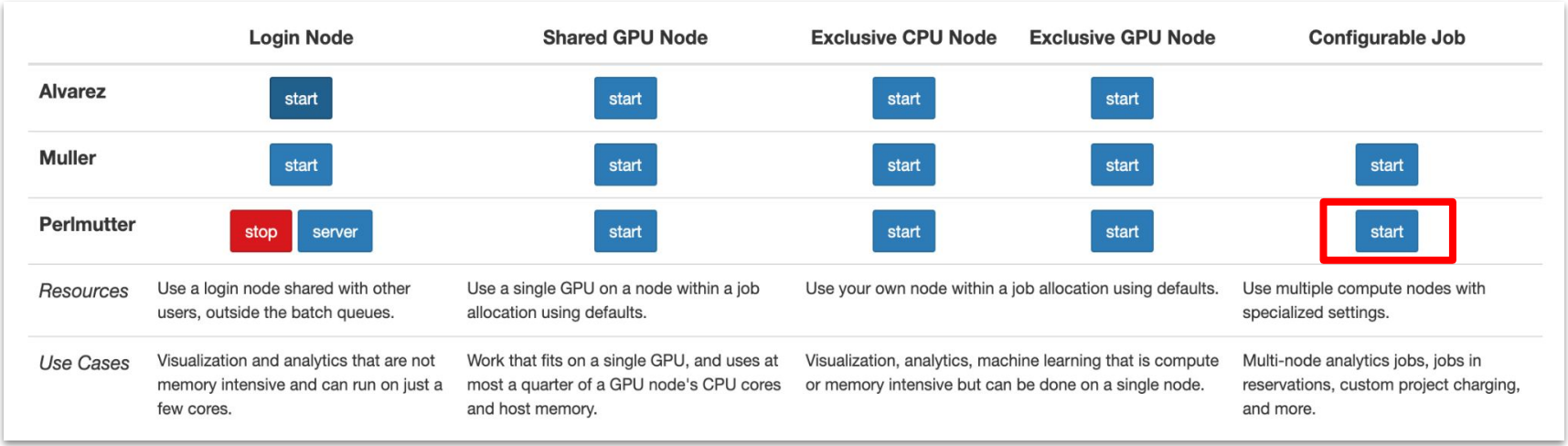
- The `versioninfo()` function can be used to check if any backends are configured correctly. Eg. for `MPI.jl` and `CUDA.jl`:

```
[3]: import MPI

[4]: MPI.versioninfo()

MPIPreferences:
  binary:  system
  abi:     MPICH
  libmpi:  libmpi_gnu_123.so
  mpiexec: srun

Package versions
  MPI.jl:             0.20.22
  MPIPreferences.jl:  0.1.11

Library information:
  libmpi:  libmpi_gnu_123.so
  libmpi dlpath:  /opt/cray/pe/lib64/libmpi_gnu_123.so
  MPI version:  3.1.0
  Library version:
    MPI VERSION    : CRAY MPICH version 8.1.28.29 (ANL base 3.4a2)
    MPI BUILD INFO : Wed Nov 15 20:57 2023 (git hash 1cde46f)
```

```
[5]: import CUDA

[6]: CUDA.versioninfo()

CUDA runtime 12.2, local installation
CUDA driver 12.2
NVIDIA driver 535.183.6

CUDA libraries:
- CUBLAS: 12.2.1
- CURAND: 10.3.3
- CUFFT: 11.0.8
- CUSOLVER: 11.5.0
- CUSPARSE: 12.1.1
- CUPTI: 2023.2.0 (API 20.0.0)
- NVML: 12.0.0+535.183.6

Julia packages:
- CUDA: 5.5.2
- CUDA_Driver_jll: 0.10.3+0
- CUDA_Runtime_jll: 0.15.3+0
- CUDA_Runtime_Discovery: 0.3.5

Toolchain:
- Julia: 1.10.4
- LLVM: 15.0.7

Preferences:
```

Using Reservations in Tutorials

- Go to https://jupyter.nersc.gov and select: "Configurable GPU" in the "Perlmutter" row



	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.

Jupyter Options:

Leave defaults, except:

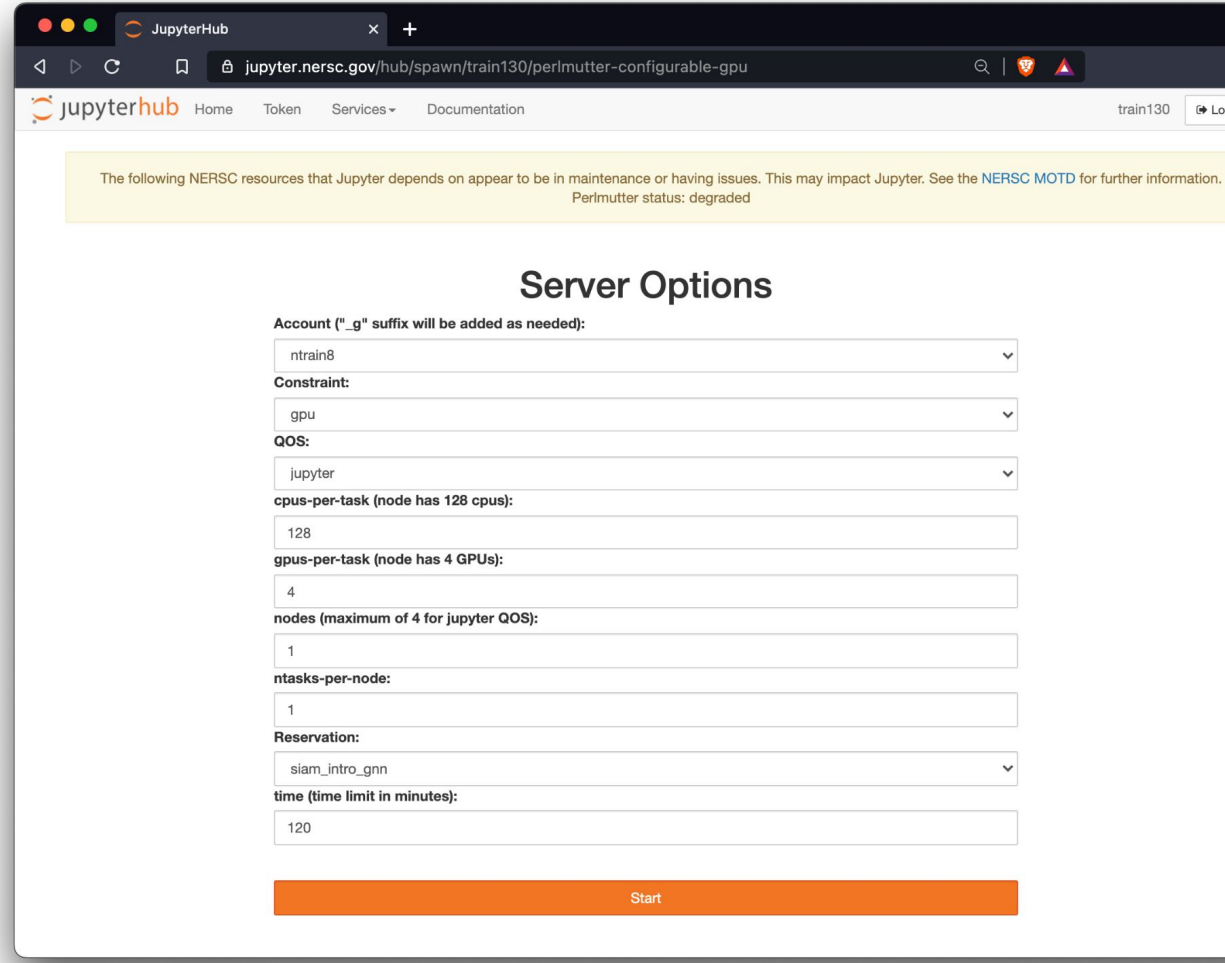
Account:

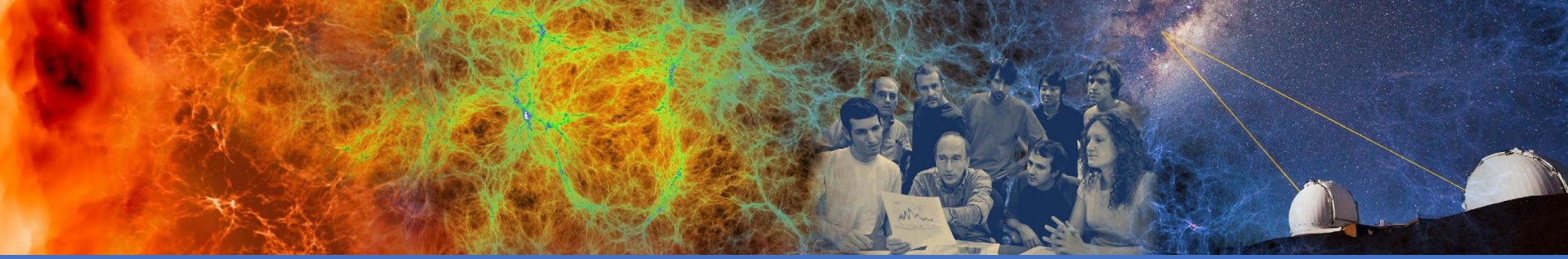
`trn019`

Reservation (leave blank)

Time

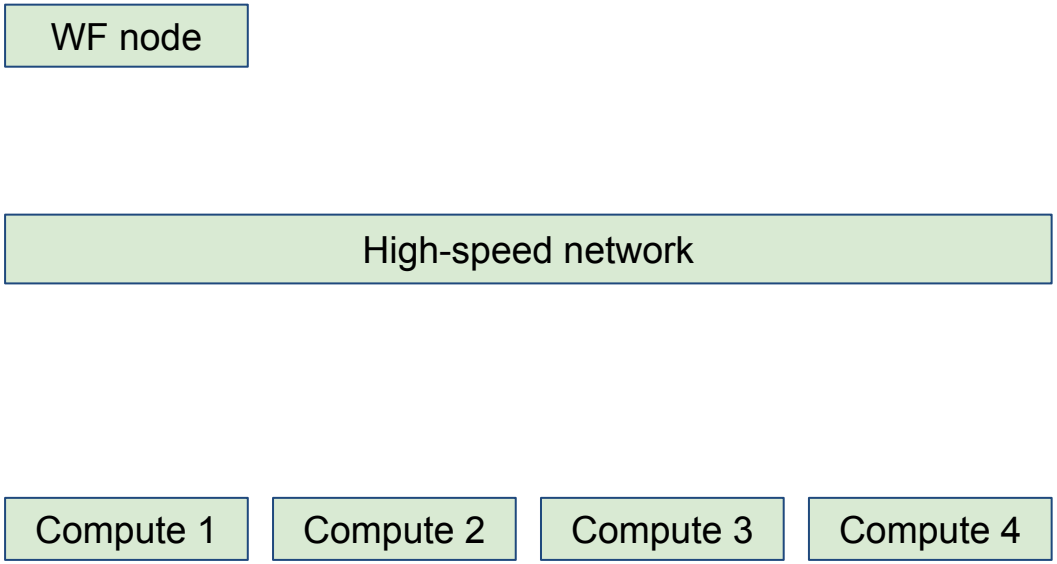
`180`



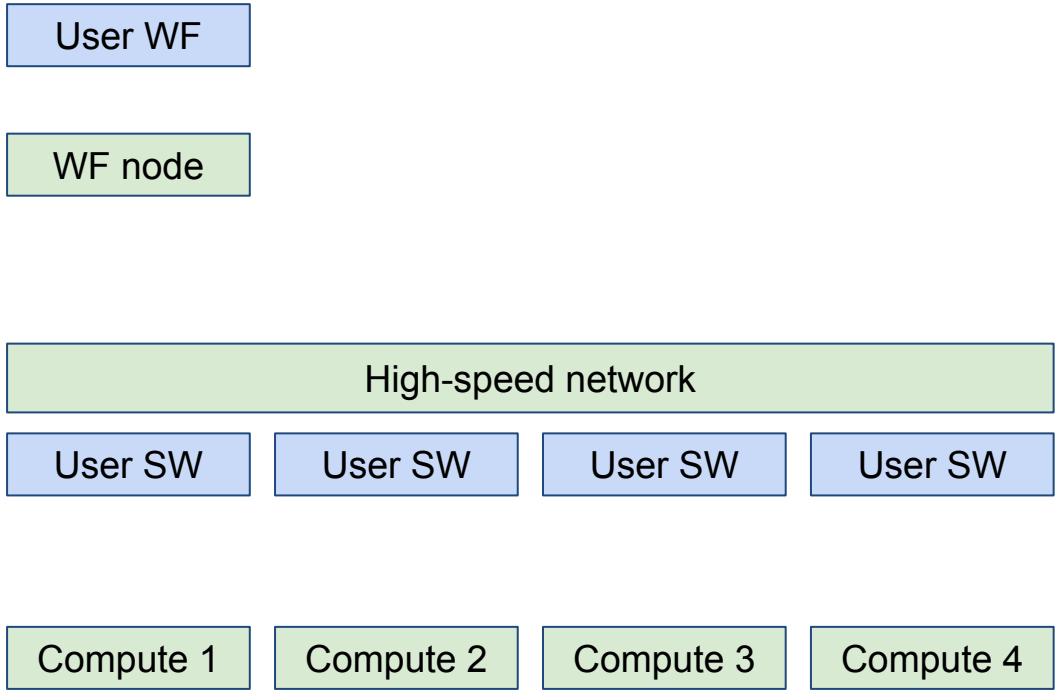


Building HPC Julia Workflows

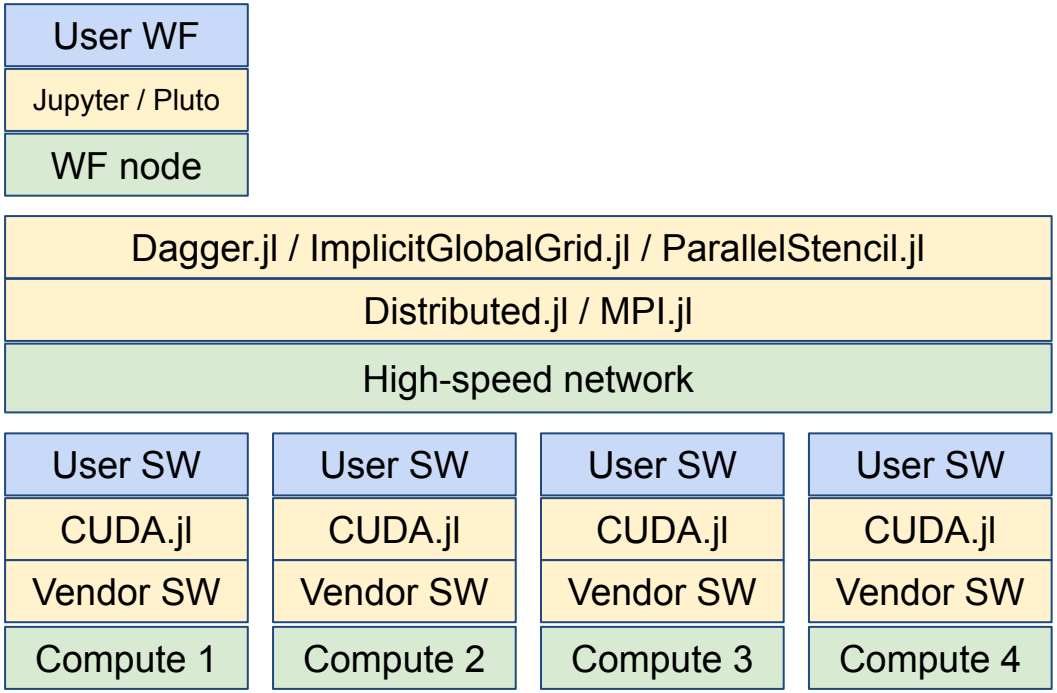
Building a HPC Workflow in Julia



Building a HPC Workflow in Julia



Building a HPC Workflow in Julia



Building a HPC Workflow in Julia