

Using Jupyter at NERSC: A Primer



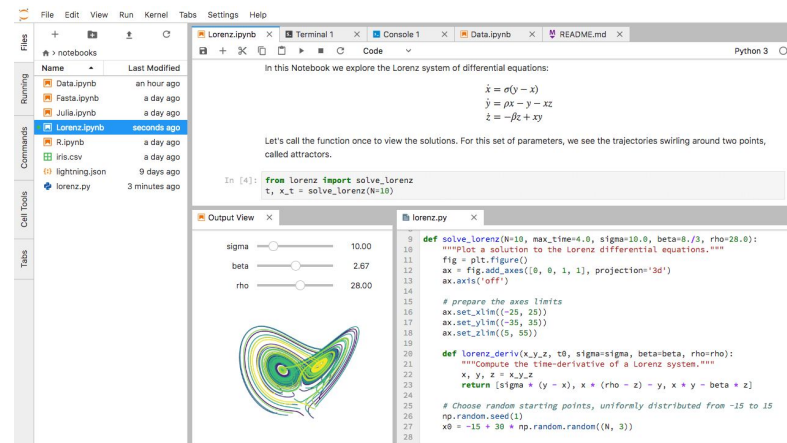
NERSC New User Training
September 8, 2023

Kelly L. Rowland, Ph.D.
User Engagement Group

What is Jupyter?

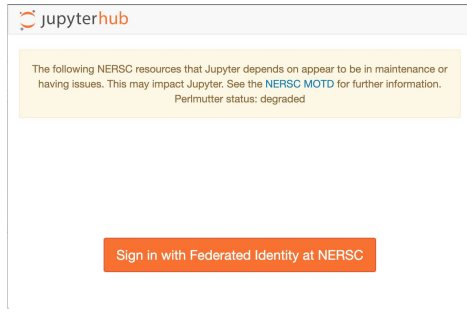


- At NERSC, we say “Jupyter” in reference to a collection of many things
 - Access shareable Jupyter “notebooks” via JupyterHub
- What can I put in a Jupyter notebook?
 - Live code
 - Equations
 - Visualizations
 - Narrative text
 - Interactive widgets
- What applications would I use a notebook for?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

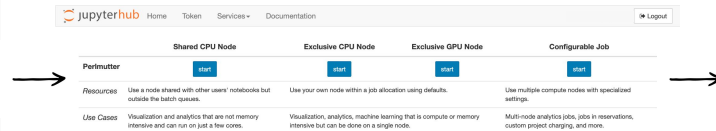


How Do I Use Jupyter at NERSC?

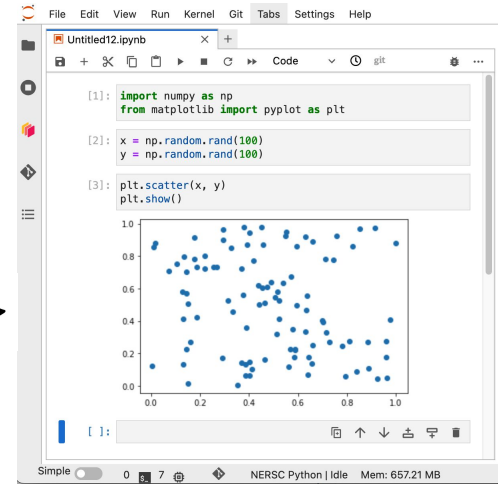
- <https://jupyter.nersc.gov>



Authenticate



Choose



Go!

How Do I Choose a Notebook Server to Spawn?

Shared CPU:

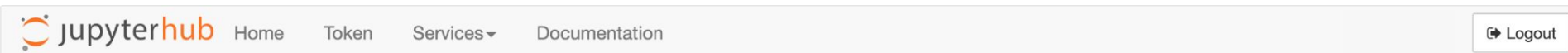
Notebook on one of 40 login nodes
Same Python env as SSH login
Can submit jobs via **!sbatch**

Exclusive CPU/GPU:

Notebook in job allocation
CPU node or GPU node
Uses NERSC hours

Configurable Job:

Notebook in job allocation
CPU node(s) or GPU node(s)
Uses NERSC hours
Can be used in reservations



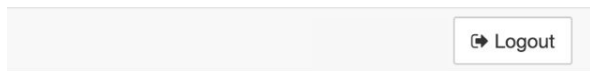
	Shared CPU Node	Exclusive CPU Node	Exclusive GPU Node	Configurable Job
Perlmutter	start	start	start	start
Resources	Use a node shared with other users' notebooks but outside the batch queues.	Use your own node within a job allocation using defaults.		Use multiple compute nodes with specialized settings.
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.

Shared = other users and processes on the same node

Exclusive and configurable = compute nodes just for your notebook and processes

Configurable Job Settings

Server Options



Configurable Job

start

Use multiple compute nodes with specialized settings.

Multi-node analytics jobs, jobs in reservations, custom project charging, and more.



Account ("_g" suffix will be added as needed):

nstaff

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:

(None)

time (time limit in minutes):

360

Start

JupyterLab Interface

The screenshot displays the JupyterLab interface. On the left, there is a sidebar with 'FAVORITES' (listing \$PSCRATCH, rowlandk, \$SCRATCH, and \$HOME) and a 'FILE BROWSER' (showing a search bar and a file list). The main area is a code editor for 'Untitled12.ipynb' in 'NERSC Python' mode. It contains three code cells:

```
[1]: import numpy as np
      from matplotlib import pyplot as plt

[2]: x = np.random.rand(100)
      y = np.random.rand(100)

[3]: plt.scatter(x, y)
      plt.show()
```

Below the code is a scatter plot with blue dots on a white background. The x and y axes both range from 0.0 to 1.0. The status bar at the bottom shows 'Simple' mode, '0' CPU usage, '2' memory usage, 'NERSC Python | Idle', 'Mem: 356.48 MB', 'Mode: Command', 'Ln 1, Col 1', and 'Untitled12.ipynb'.

JupyterLab Interface: NERSC Add-ons

- Favorites
- Bookmark your favorite places on the file systems
- Pre-populated with \$HOME and \$PSCRATCH
- Add the current directory by clicking the ★ icon

The screenshot displays the JupyterLab interface with the following components:

- Menu Bar:** File, Edit, View, Run, Kernel, Git, Tabs, Settings, Help.
- File Browser:** Shows a search bar and a list of files in the directory `/.../r/rowlandk/`. The files are listed with their names and last modified times.
- Favorites Panel:** A red box highlights the Favorites section, which includes `$PSCRATCH`, `rowlandk`, `$SCRATCH`, and `$HOME`.
- Code Editor:** Contains three code cells:

```
[1]: import numpy as np
      from matplotlib import pyplot as plt

[2]: x = np.random.rand(100)
      y = np.random.rand(100)

[3]: plt.scatter(x, y)
      plt.show()
```
- Figure:** A scatter plot showing 100 data points (blue dots) scattered across a 2D plane with both axes ranging from 0.0 to 1.0.
- Status Bar:** Shows "Simple" mode, 0 files, 2 tabs, and system information: "NERSC Python | Idle Mem: 356.48 MB Mode: Command Ln 1, Col 1 Untitled12.ipynb".

JupyterLab Interface: NERSC Add-ons

- **Open from Path...**
- **Jump to anywhere** in the file system

- **Recents**
- **Recent locations** you've visited on the file system

The screenshot displays the JupyterLab interface. The top menu bar includes File, Edit, View, Run, Kernel, Git, Tabs, Settings, and Help. The File menu is open, with options: New, New Launcher, Open from Path..., Open from URL..., Recents, and Favorites. The File Browser on the left shows a search bar and a list of files under the path / ... / r / rowlandk /. The main area contains a code editor with the following Python code:

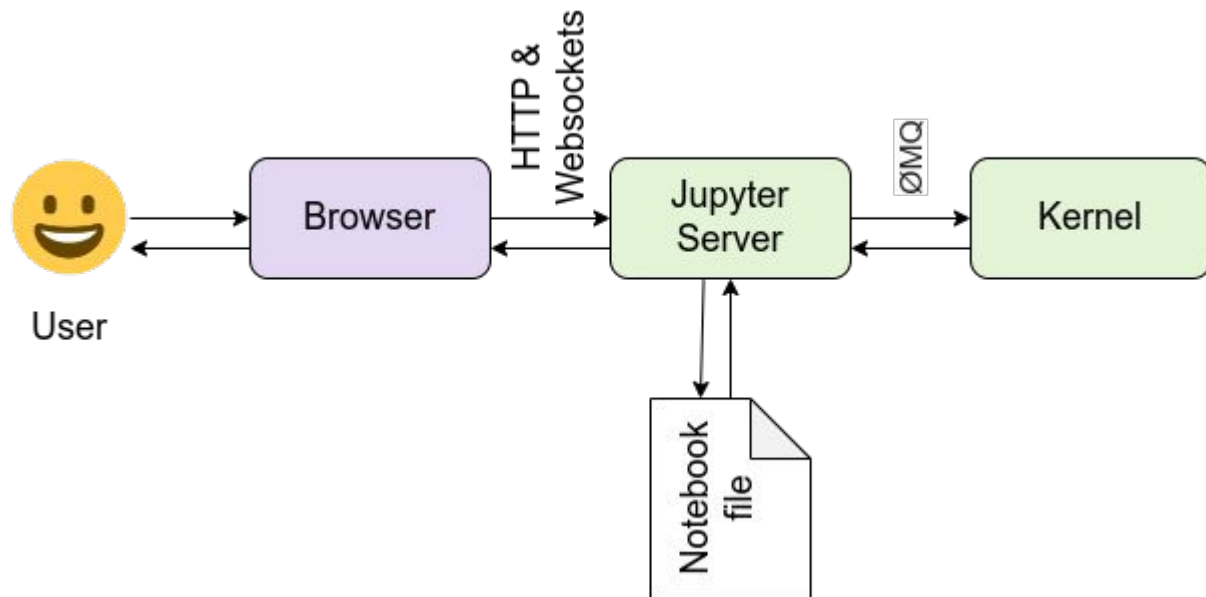
```
import numpy as np
from matplotlib import pyplot as plt

x = np.random.rand(100)
y = np.random.rand(100)

plt.scatter(x, y)
plt.show()
```

Below the code is a scatter plot showing 100 blue data points distributed across a square area with axes ranging from 0.0 to 1.0. The status bar at the bottom indicates 'Simple' mode, 0 files, 2 tabs, NERSC Python | Idle, Mem: 356.48 MB, Mode: Command, Ln 1, Col 1, and the file name 'Untitled12.ipynb'.

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, R
 - ML packages
- Bring your own kernel

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

Your Own Jupyter Kernel

- A common Jupyter question:
 - [“How do I take a conda environment and use it from Jupyter?”](#)
- Several ways to accomplish this; we recommend:

```
$ module load python
$ conda create -n myenv python=3.9
$ source activate myenv
(myenv) $ conda install ipykernel <other-packages> ...
(myenv) $ python -m ipykernel install --user --name myenv-jupyter
```

- Point your browser to jupyter.nersc.gov
 - May need to restart notebook server via control panel
- Kernel “myenv-jupyter” should be present in the kernel list

 This creates a “kernelpec” file

The kernelspec File

```
(myenv) user@login01:~$ cat \  
    $HOME/.local/share/jupyter/kernels/myenv-jupyter/kernel.json  
{  
  "argv": [  
    "/global/homes/u/user/.conda/envs/myenv/bin/python",  
    "-m",  
    "ipykernel_launcher",  
    "-f",  
    "{connection_file}"  
  ],  
  "display_name": "myenv-jupyter",  
  "language": "python"  
}
```

Additional Customization

```
{
  "argv": [
    "/global/homes/u/user/.conda/envs/myenv/bin/python",
    "-m",
    "ipykernel_launcher",
    "-f",
    "{connection_file}"
  ],
  "display_name": "myenv-jupyter",
  "language": "python",
  "env": {
    "PATH": ...,
    "LD_LIBRARY_PATH": ...,
  }
}
```

Additional Customization - Kernel Helper Script

```
{  
  "argv": [  
    "/global/homes/u/user/kernel-helper.sh",  
    "-f",  
    "{connection_file}"  
  ],  
  "display_name": "myenv-jupyter2",  
  "language": "python",  
}
```

The kernel helper script is the most flexible approach for NERSC users since it easily enables use of modules, environment variables, etc.

Meanwhile, in kernel-helper.sh:

```
#!/bin/bash  
export SOMETHING=123  
module load foo  
exec python -m ipykernel "$@"
```

A Shifter kernelspec File

```
{  
  "argv": [  
    "shifter",  
    "--image=continuumio/anaconda3:latest",  
    "/opt/conda/bin/python",  
    "-m",  
    "ipykernel_launcher",  
    "-f",  
    "{connection_file}"  
  ],  
  "display_name": "my-shifter-kernel",  
  "language": "python"  
}
```

Image name

Path to Python in the image

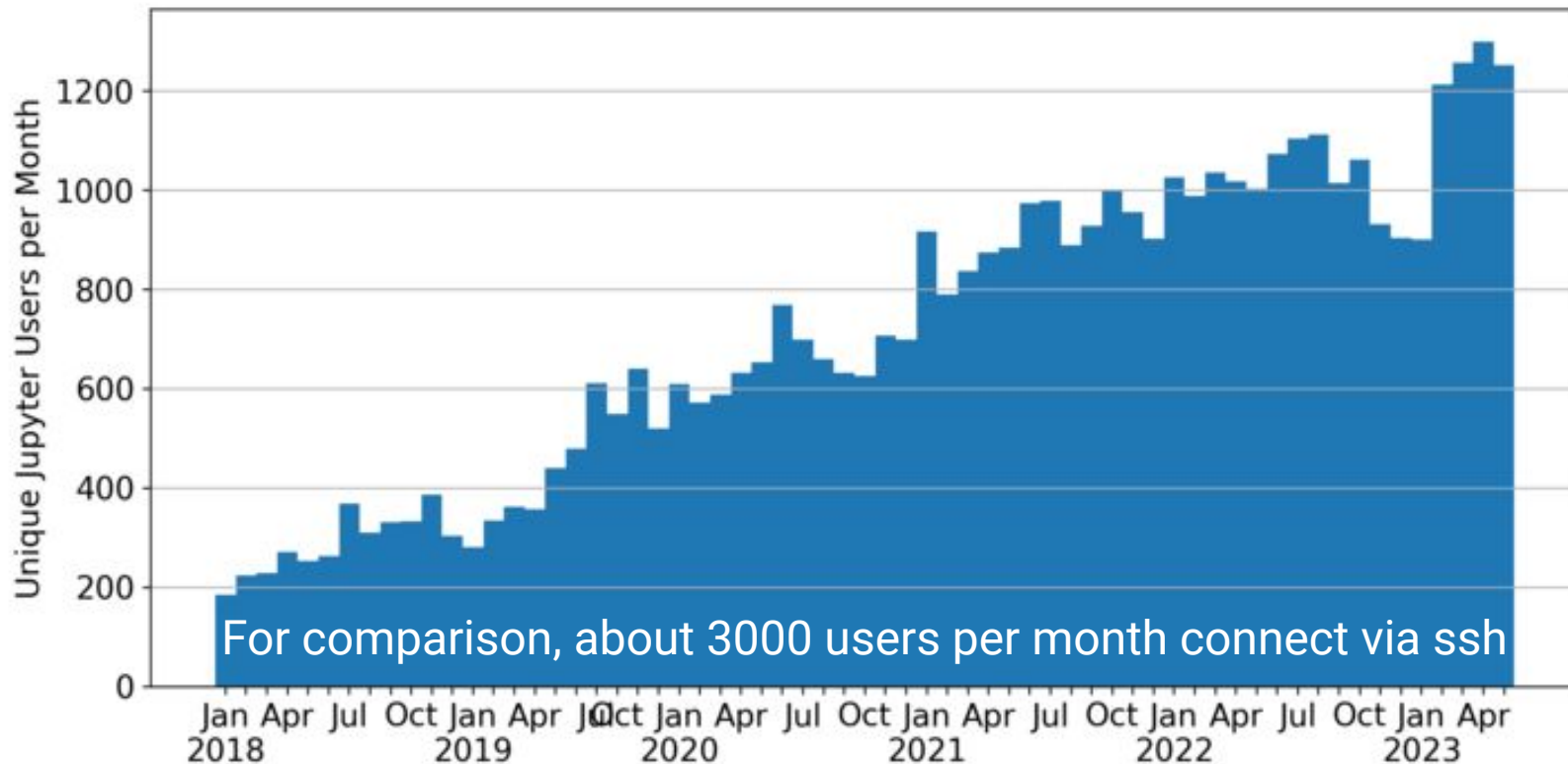


Debugging Jupyter Issues

```
(myenv) user@login01:~$ cat ~/.jupyter-perlmutter.log
```

```
[IPKernelApp] ERROR | No such comm target registered: jupyter.widget.control
[IPKernelApp] WARNING | No such comm: aa07e0e8-5f78-4899-ab3f-8af339f1318e
[W 2023-06-12 14:20:16.974 SingleUserLabApp zmqhandlers:227] WebSocket ping timeout after 119813 ms.
[I 2023-06-12 14:20:16.977 SingleUserLabApp kernelmanager:321] Starting buffering for
04d30821-f7f4-46c2-a016-ba576b5af07c:74105d47-601c-4d77-8316-c75fdfae4bab
[W 2023-06-12 14:20:17.035 SingleUserLabApp zmqhandlers:227] WebSocket ping timeout after 119874 ms.
[I 2023-06-12 14:20:17.036 SingleUserLabApp kernelmanager:321] Starting buffering for
fcb31e09-6a2a-427e-aaf8-f15d1a443bda:fbe5d17f-91a2-49d7-bf22-1da23dc8ef4b
[W 2023-06-12 14:20:17.110 SingleUserLabApp zmqhandlers:227] WebSocket ping timeout after 119949 ms.
[I 2023-06-12 14:20:17.111 SingleUserLabApp kernelmanager:321] Starting buffering for
fac60c02-f294-4a49-b711-89501fefcfe8:006691d0-c3c5-480c-aacb-ffde01ab6169
[W 2023-06-12 14:20:17.176 SingleUserLabApp zmqhandlers:227] WebSocket ping timeout after 119988 ms.
[I 2023-06-12 14:20:17.177 SingleUserLabApp kernelmanager:321] Starting buffering for
19490e67-80b6-4745-85cb-0d5b8411c959:dc46ed9a-1d6e-4142-a567-c4ad9aalea3d
[W 2023-06-12 14:20:17.288 SingleUserLabApp zmqhandlers:227] WebSocket ping timeout after 120040 ms.
[I 2023-06-12 14:20:17.290 SingleUserLabApp handlers:454] Restoring connection for
04d30821-f7f4-46c2-a016-ba576b5af07c:74105d47-601c-4d77-8316-c75fdfae4bab
[I 2023-06-12 14:20:17.291 SingleUserLabApp kernelmanager:321] Starting buffering for
b9cb4f21-1f8c-4917-b7a5-4653b158d87b:230a9755-8454-4f84-a097-041c7e88b5bb
[IPKernelApp] ERROR | No such comm target registered: jupyter.widget.control
[IPKernelApp] WARNING | No such comm: 8844d734-bdf7-4159-b1ab-4534db8105b6
[W 2023-06-12 14:20:17.368 SingleUserLabApp zmqhandlers:227] WebSocket ping timeout after 119844 ms.
```

Jupyter Usage at NERSC



Jupyter at NERSC - Summary

- Go to <https://jupyter.nersc.gov> to use Jupyter at NERSC
- Use a kernelspec to use a conda environment in your notebook
- You can customize those kernelspec files in many ways
- We work on making Jupyter work and work better for you

- Always looking for:
 - New ways to empower Jupyter users
 - Feedback, advice, and even help: <https://help.nersc.gov/>

Thank you!

