UC **SANTA BARBARA**

CSC Center for Scientific Computing

# HPC Workshop 2

Feb. 18, 2024

11:30 – 12:30 pm (followed by pizza)

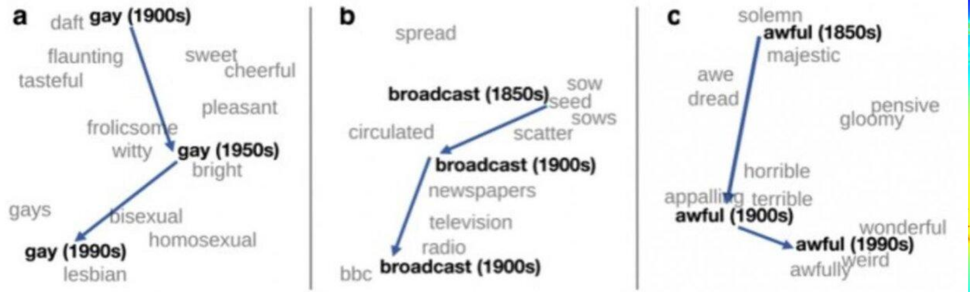
Location: Elings Hall 1601

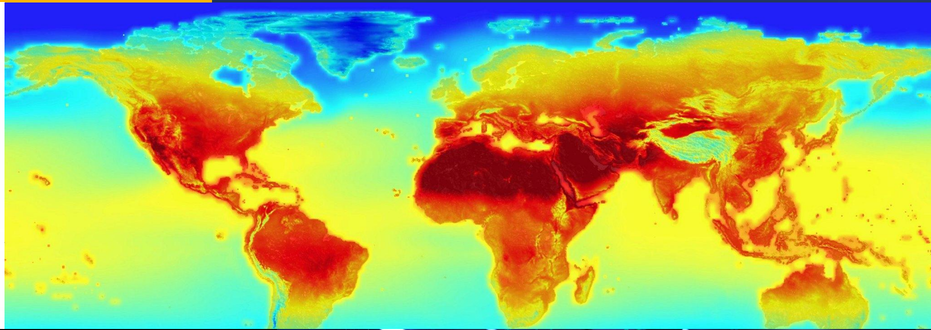
Register @ https://csc.cnsi.ucsb.edu

Quickly start using HPC resource at UCSB

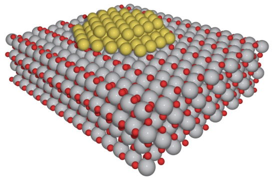
- SLURM Array and Job Dependency
- Running Jupyter Notebook/Lab and VS Code on the Cluster
- NSF ACCESS allocation
- National & Commercial Cloud Computing Resources

Computational Linguistics



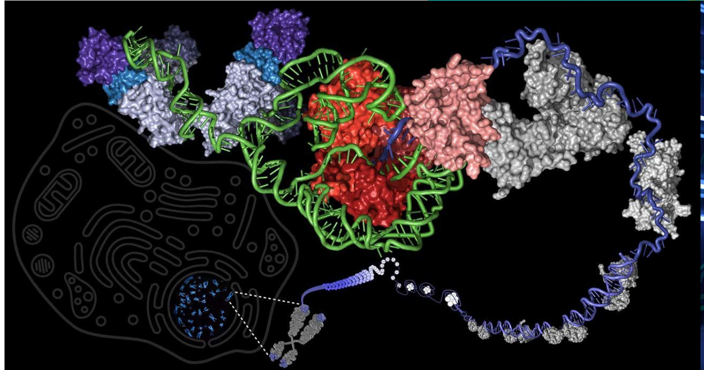


KS-DFT



The total energy in Kohn-Sham Density Functional Theory (KS-DFT) is expressed as

$$E_{total} = \underbrace{T_s}_{\text{Kohn-Sham kinetic energy}} + \underbrace{\int d\mathbf{r} V_{ext}(\mathbf{r})\rho(\mathbf{r})}_{\text{External energy}} + \underbrace{E_{xc}[\rho]}_{\text{XC energy}} + \underbrace{\frac{1}{2}\int\int d\mathbf{r}d\mathbf{r}' \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}}_{\text{Hartree energy}}$$



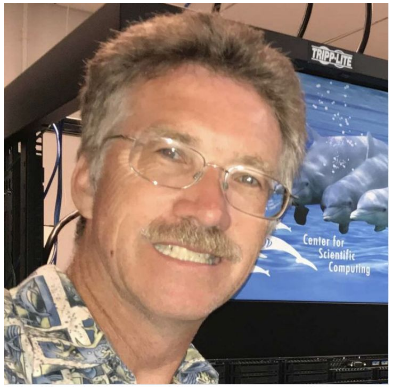
UC **SANTA BARBARA**

# Introduction to High-Performance Computing (HPC)

Paul Weakliem, Fuzzy Rogers, and Jay Chi

February 18, 2025

# Our Team



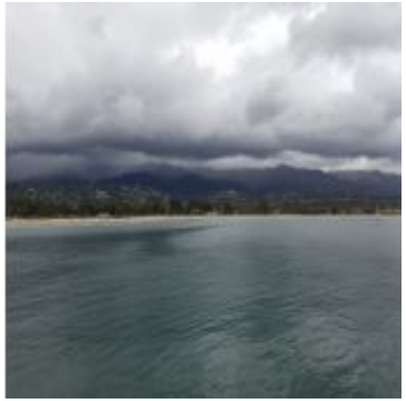
Paul Weakliem, PhD
Co-Director
Center for Scientific Computing &
California Nanosystems Institute
Eling 3231
weakliem@cnsi.ucsb.edu

Fuzzy Rogers
Research Computing Administrator
Center for Scientific Computing &
Materials Research Laboratory
MRL 2066B
fuz@mrl.ucsb.edu

Yu-Chieh "Jay" Chi, PhD
Research Computing Consultant
Center for Scientific Computing &
Enterprise Technology Services
Elings 3213
jaychi@ucsb.edu

CSC Center for Scientific Computing CNSI MRL UC SANTA BARBARA Information Technology

## Our Research IT Partners



Letters & Science
INFORMATION TECHNOLOGY

Justin Maness
Interim Director of Engineering Computing Infrastructure
3152a Harold Frank Hall
jmanes@engineering.ucsb.edu

Michael Colee
Director of Research IT
6703 Ellison Hall
mtc@eri.ucsb.edu

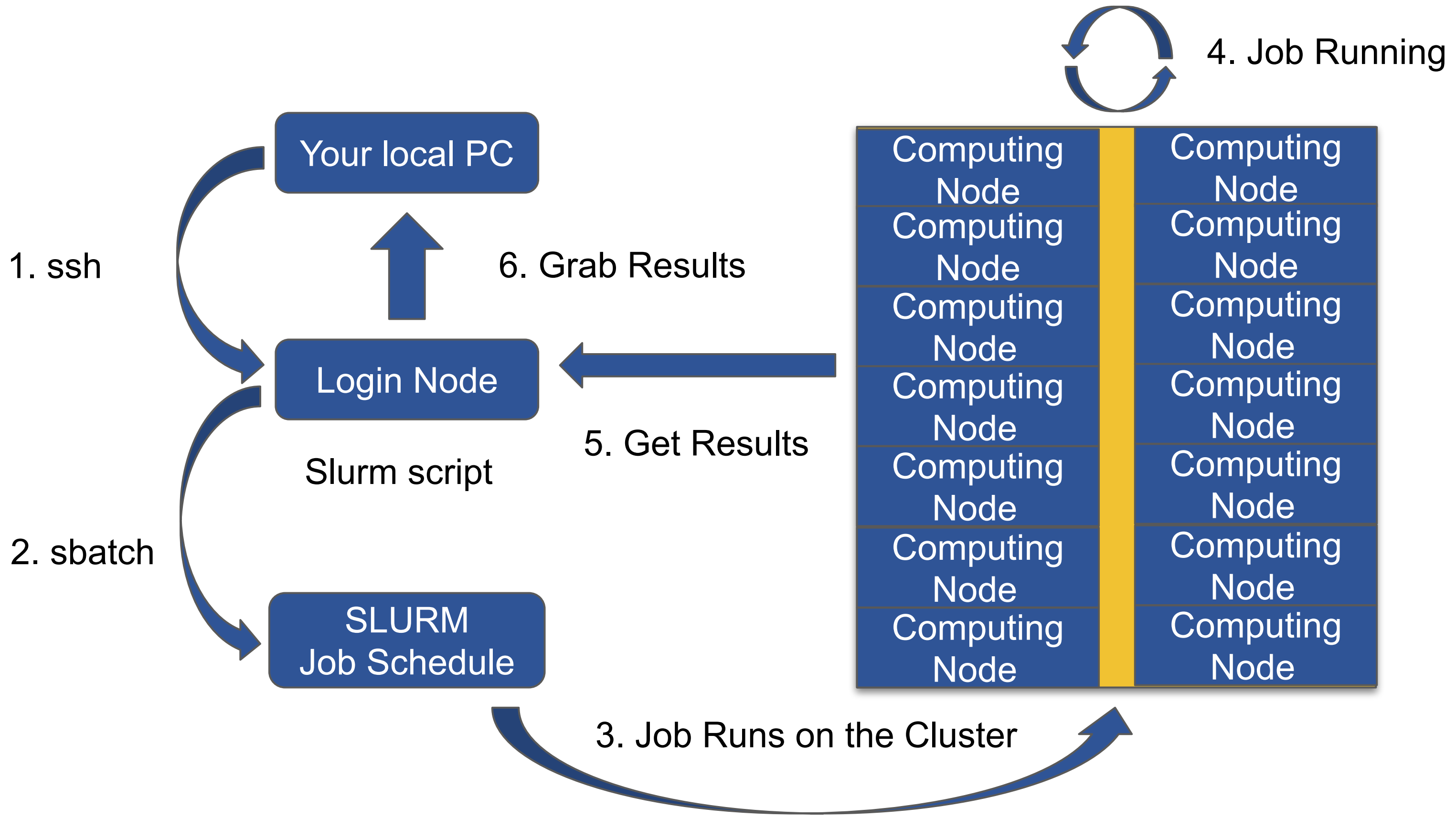
Ted Cabeen
Director of Life Science Computing Group (LSCG)
2306 Life Science
ted.cabeeen@lscg.ucsb.edu

Andreas Boschke
Director of Infor. Infrastructure at Letter & Science IT (LSIT)
2306 Life Science
andreas@lsit.ucsb.edu

## Agenda

- HPC Workflow
- SLURM Array
- SLURM Job Dependency
- Running Jupyter Notebook/Lab and VS Code on the cluster
    - Running Jupyter Notebook/lab on the POD/Braid2
    - Running VS Code on the POD/Braid2
    - Google Colab
- Introduction to National HPC/Supercomputer resources
    - ACCESS allocation
    - Regional Computing Resource (SDSC)
    - Cloud Computing: Jetstream2 from Indiana University

# General HPC Workflow



## Example Slurm Job Submission script

Slurm job script file: job.s

```
#!/bin/bash                          ### Set linux shell: Telling the shell to run the script using the batch
#SBATCH -J 'testJob'                 ### Job Name
#SBATCH --nodes=1                    ### No. of Nodes
#SBATCH --ntasks=1                   ### No. of Tasks
#SBATCH -p gpu                       ### Submit the job to Partition
#SBATCH –gres=gpu:1                  ### Request 1 GPU
#SBATCH -o outLog                    ### Output Log File (Optional)
#SBATCH -e errLog                    ### Error Log File (Optional but suggest to have it)
#SBATCH -t 00:10:00                  ### Job Execution Time
#SBATCH --mail-user=usernam@ucsb.edu ### Mail to you (Optional)
#SBATCH --mail-type ALL              ### Mail send you when the job starts and end (Optional)

module purge all
module load anaconda                 ### Load softwares that the job depends on to execute
source activate my_env_1

cd $SLURM_SUBMIT_DIR/                ### Absolute path of the current working directory when you submit the job

python my_python.py
```

# Job Arrays

- According to the Slurm Workload Manager, "Job arrays offer a mechanism for submitting and managing collections of similar jobs quickly and easily, … . **All jobs must have the same initial options** (e.g., size, time limit, etc.)"
- In general, job arrays are useful for applying the **same processing routine** to a collection of **multiple input data files**. Job arrays offer a very simple way to submit a large number of independent processing jobs.

```
#!/bin/bash
#SBATCH -J 'slurmArray'                 ### Job Name
#SBATCH --nodes=1                       ### No. of Nodes
#SBATCH --ntasks=1                      ### No. of Tasks
#SBATCH -p short                        ### Submit the job to Partition (Optional)
#SBATCH -o outLog_%A_%a                 ### Output Log File (Optional)
#SBATCH -e errLog_%A_%a                 ### Error Log File (Optional but suggest to have it)
#SBATCH -t 00:10:00                     ### Job Execution Time
#SBATCH –array=0-3
#SBATCH --mail-user=usernam@ucsb.edu    ### Mail to you (Optional)
#SBATCH --mail-type ALL                 ### Mail send you when the job starts and end (Optional)
```

## Slurm Job Array Submission script

- The %A_%a construct in the output and error file names is used to generate unique output and error files based on the master job ID (%A) and the array-tasks ID (%a).
- Job Array indices can be specified array index values, a range of index values, and an optional step size.

```
# Submit a job array with index values between 0 and 15
#SBATCH –array=0-15

# Submit a job array with index values of 1, 3, 9, and 15
#SBATCH –array=1, 3, 9, 15

# Submit a job array with index values between 1 and 16 with a step size of 2
#SBATCH –array=1-16:2
```

## Slurm Job Array Submission script

```
#!/bin/bash
#SBATCH -J 'slurmArray’
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH -p short
#SBATCH -o outLog_%A_%a
#SBATCH -e errLog_%A_%a
#SBATCH -t 00:10:00
#SBATCH –array=0-3

echo "SLURM_JOB_ID: " $SLURM_JOBID
echo "SLURM_ARRAY_JOB_ID: " $SLURM_ARRAY_JOB_ID
echo "SLURM_ARRAY_TASK_ID: " $SLURM_ARRAY_TASK_ID
echo "SLURM_ARRAY_TASK_COUNT: " $SLURM_ARRAY_TASK_COUNT
echo "SLURM_ARRAY_TASK_MAX: " $SLURM_ARRAY_TASK_MAX
echo "SLURM_ARRAY_TASK_MIN: " $SLURM_ARRAY_TASK_MIN
```

```
SLURM_JOB_ID:  3405632
SLURM_ARRAY_JOB_ID:  3405629
SLURM_ARRAY_TASK_ID:  0
SLURM_ARRAY_TASK_COUNT:  4
SLURM_ARRAY_TASK_MAX:  3
SLURM_ARRAY_TASK_MIN:  0

SLURM_JOB_ID:  3405633
SLURM_ARRAY_JOB_ID:  3405629
SLURM_ARRAY_TASK_ID:  1
SLURM_ARRAY_TASK_COUNT:  4
SLURM_ARRAY_TASK_MAX:  3
SLURM_ARRAY_TASK_MIN:  0

SLURM_JOB_ID:  3405629
SLURM_ARRAY_JOB_ID:  3405629
SLURM_ARRAY_TASK_ID:  3
SLURM_ARRAY_TASK_COUNT:  4
SLURM_ARRAY_TASK_MAX:  3
SLURM_ARRAY_TASK_MIN:  0
```

## Dependency Jobs

- You can schedule jobs depending on the termination status of previously scheduled jobs. This way, you can concatenate your jobs into a pipeline or expand to more complicated dependencies.
- For example, job1.s is a submission script you plan to submit a batch job:

```
#!/bin/bash
#SBATCH -J 'JobDep1'                         ### Job Name
#SBATCH --nodes=1                            ### No. of Nodes
#SBATCH --ntasks=1                           ### No. of Tasks
#SBATCH -p short                             ### Submit the job to Partition (Optional)
#SBATCH -o outLog_%x_%j                      ### Output Log File (Optional)
#SBATCH -e errLog_%x_%j                      ### Error Log File (Optional but suggest to have it)
#SBATCH -t 00:10:00                          ### Job Execution Time
#SBATCH --mail-user=usernam@ucsb.edu         ### Mail to you (Optional)
#SBATCH --mail-type ALL                      ### Mail send you when the job starts and end (Optional)

# Run Bash Command
echo "***** My first Program *****"
echo "***** Prepare the Data *****"
echo "*****Done for Parparation *****"
echo "Time: " $(date +"%T")
```

## Dependency Jobs

- Submit the job script to the Slurm job scheduler from the POD login node:

```
$ sbatch job1.s
Submitted batch job 1234567
```

- You can submit another job that is put on the waiting list of the queue.

```
$ sbatch -dependency=aftercorr:1234567 job2.s
```

- This command indicates that job2.s will be put in the queue after the job ID 1234567 is terminated for any reason. The *dependency option flag* can be after, afterany, aftercorr, afterok, and afternotok.
- The following command would submit 2 jobs with respect to their dependencies.

```
# First Job
jobID_1=$(sbatch job1.s | cut -f 4 -d' ')

# Second Job - this job depends on the first job
sbatch --dependency=aftercorr:$jobID_1 job2.s
```

## Slurm Job Dependency Submission script

| | |
|---|---|
| after | This job is execution after the specified jobs have begun execution |
| afterany | This job can begin execution after the specified jobs have been terminated |
| aftercorr | A task of this job array can begin execution after the corresponding task ID in the specified job has completed successfully |
| afternotok | This job can begin execution after the specified jobs have terminated in some failed state |
| afterok | This job can begin execution after the specified jobs have been successfully executed |
| singleton | This job can begin execution after any previously launched jobs sharing the same job name and the user has terminated |

## Slurm Job Dependency Submission script

Slurm job script file: job1.s

```
#!/bin/bash
#SBATCH -J 'JobDep1'                         ### Job Name
#SBATCH --nodes=1                            ### No. of Nodes
#SBATCH --ntasks=1                           ### No. of Tasks
#SBATCH -p short                             ### Submit the job to Partition (Optional)
#SBATCH -o outLog_%x_%j                      ### Output Log File (Optional)
#SBATCH -e errLog_%x_%j                      ### Error Log File (Optional but suggest to have it)
#SBATCH -t 00:10:00                          ### Job Execution Time
#SBATCH --mail-user=usernam@ucsb.edu         ### Mail to you (Optional)
#SBATCH --mail-type ALL                      ### Mail send you when the job starts and end (Optional)

# Run Bash Command
echo "***** My first Program *****"
echo "***** Prepare the Data *****"
echo "***** Done for Parparation *****"
echo "Time: " $(date +"%T")
```

## Slurm Job Dependency Submission script

Slurm job script file: job2.s

```
#!/bin/bash
#SBATCH -J 'JobDep2'                              ### Job Name
#SBATCH --nodes=1                                 ### No. of Nodes
#SBATCH --ntasks=1                                ### No. of Tasks
#SBATCH -p batch                                  ### Submit the job to Partition (Optional)
#SBATCH -o outLog_%x_%j                           ### Output Log File (Optional)
#SBATCH -e errLog_%x_%j                           ### Error Log File (Optional but suggest to have it)
#SBATCH -t 00:10:00                               ### Job Execution Time
#SBATCH --mail-user=usernam@ucsb.edu              ### Mail to you (Optional)
#SBATCH --mail-type ALL                           ### Mail send you when the job starts and end (Optional)

# Run Bash Command
echo "***** Start the Program *****"
echo "***** Code1 is Running *****"
echo "***** Code2 is Running *****"
echo "***** End the Program *****"
echo "Time: " $(date +"%T")
```

# Slurm Job Dependency Submission script

Slurm job script file: job3.s

```
#!/bin/bash
#SBATCH -J 'JobDep3'                       ### Job Name
#SBATCH --nodes=1                          ### No. of Nodes
#SBATCH --ntasks=1                         ### No. of Tasks
#SBATCH -p short                           ### Submit the job to Partition (Optional)
#SBATCH -o outLog_%x_%j                    ### Output Log File (Optional)
#SBATCH -e errLog_%x_%j                    ### Error Log File (Optional but suggest to have it)
#SBATCH -t 00:10:00                        ### Job Execution Time
#SBATCH --mail-user=usernam@ucsb.edu       ### Mail to you (Optional)
#SBATCH --mail-type ALL                    ### Mail send you when the job starts and end (Optional)

# Run Bash Command
echo "***** The Last Step *****"
echo "***** Analyze the Data *****"
echo "***** Done for analyzing data *****"
echo "Time: " $(date +"%T")
```

## Slurm Job Dependency Submission script

batch script file: depJOB.s

```
#!/bin/bash

# First Job
jobID_1=$(sbatch job1.s | cut -f 4 -d' ')

# Second Job - this job depends on the first job
jobID_2=$(sbatch --dependency=aftercorr:$jobID_1 job2.s | cut -f 4 -d' ')

# Third Job - this job also depends on the second job
sbatch --dependency=aftercorr:$jobID_2 job3.s
```

- Execute the batch job script from the POD login node:

```
$ sh depJOB.s
Submitted batch job 1234567
```

## Why Run Jupyter Notebook/Lab, VS Code on the cluster?

- Computational resource requirement (GPU, multiple Cores, and etc.)
- Large memory requirement for your data
- Centralized the Large Data Storage
    - Convenience to analyze your Large scale data on the cluster
    - Convenience to share the large data with research group
- Friendly user interface
- Scaling up to long runtimes

**However - this is not what HPC cluster (e.g. Pod) is designed for!!!**

*Coming soon for UCSB - https://ucsb-csc.nrp-nautilus.io (National Research Platform)*

## Running Interactive Job

- Interactive computing refers to software which accepts input from the user as it runs. **Interactive computing** involves *real-time* user inputs to perform tasks on a set of compute node(s) including:
    - Code development, real-time data exploration, etc.
    - Used when applications have large data sets or are too large to download to local device, or too large to compute on the local device
    - Actions performed on remote compute nodes as a result of user input or program out.
- To request an interactive computing node with 4 cores for 4 hours:

```
$ srun -N 1 -n 4 -p batch --time=4:00:00 --pty bash -i
```

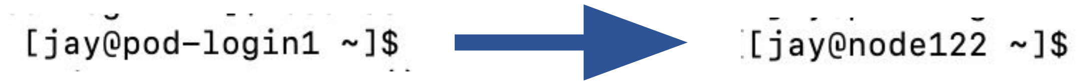
- To request an interactive computing GPU node for 4 hours:

```
$ srun -N 1 -n 1 -p gpu --gres=gpu:1 --time=4:00:00 --pty bash -i
```

# Set Up Your Jupyter Notebook on the POD/Braid2

- Get to a compute node from the login node

```
$ srun -N 1 -n 1 -p gpu  --gres=gpu:1 --time=04:00:00 --pty bash -i
```



```
[jay@pod-login1 ~]$
```

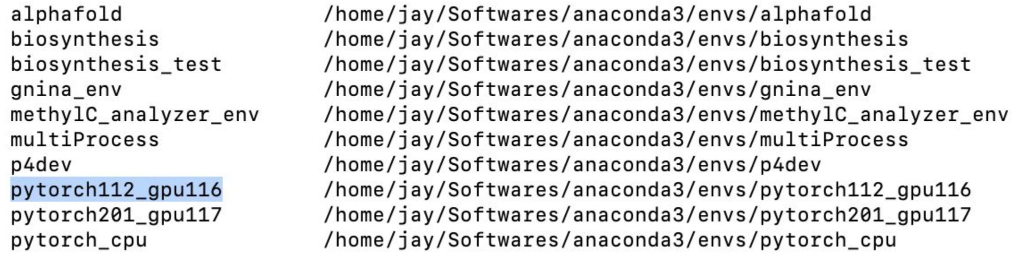
```
[jay@node122 ~]$
```

- Make sure your conda environment is activated

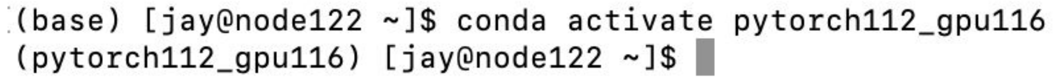
```
(base) [jay@node122 ~]$
```

- Activate the specific conda environment

```
$ conda activate pytorch112_gpu116
```



```
alphafold               /home/jay/Softwares/anaconda3/envs/alphafold
biosynthesis            /home/jay/Softwares/anaconda3/envs/biosynthesis
biosynthesis_test       /home/jay/Softwares/anaconda3/envs/biosynthesis_test
gnina_env               /home/jay/Softwares/anaconda3/envs/gnina_env
methylC_analyzer_env    /home/jay/Softwares/anaconda3/envs/methylC_analyzer_env
multiProcess            /home/jay/Softwares/anaconda3/envs/multiProcess
p4dev                   /home/jay/Softwares/anaconda3/envs/p4dev
pytorch112_gpu116       /home/jay/Softwares/anaconda3/envs/pytorch112_gpu116
pytorch201_gpu117       /home/jay/Softwares/anaconda3/envs/pytorch201_gpu117
pytorch_cpu             /home/jay/Softwares/anaconda3/envs/pytorch_cpu
```

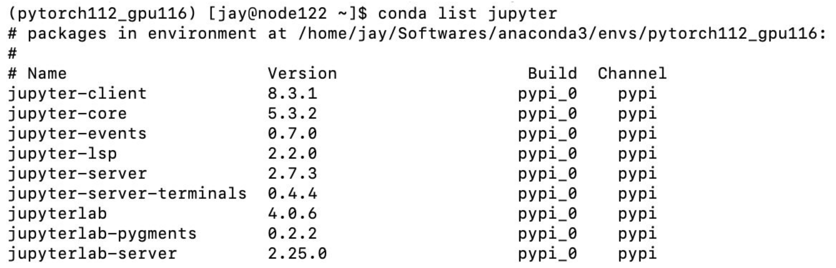


```
(base) [jay@node122 ~]$ conda activate pytorch112_gpu116
(pytorch112_gpu116) [jay@node122 ~]$ 
```

## Set Up Your Jupyter Notebook on the POD/Braid2

- Make sure the jupyter has been installed in the conda environment

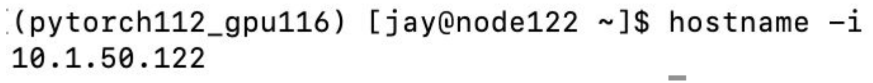
$ conda list jupyter



```
(pytorch112_gpu116) [jay@node122 ~]$ conda list jupyter
# packages in environment at /home/jay/Softwares/anaconda3/envs/pytorch112_gpu116:
#
# Name                    Version                   Build  Channel
jupyter-client            8.3.1                    pypi_0    pypi
jupyter-core              5.3.2                    pypi_0    pypi
jupyter-events            0.7.0                    pypi_0    pypi
jupyter-lsp               2.2.0                    pypi_0    pypi
jupyter-server            2.7.3                    pypi_0    pypi
jupyter-server-terminals  0.4.4                    pypi_0    pypi
jupyterlab                4.0.6                    pypi_0    pypi
jupyterlab-pygments       0.2.2                    pypi_0    pypi
jupyterlab-server         2.25.0                   pypi_0    pypi
```

- Get the ip from the host

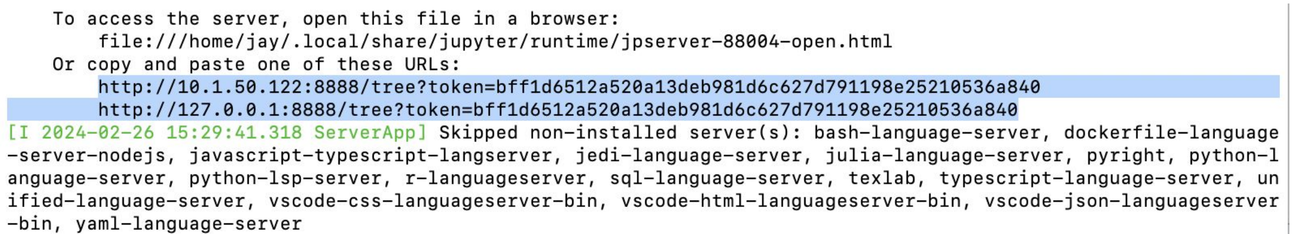
$ hostname -i



```
(pytorch112_gpu116) [jay@node122 ~]$ hostname -i
10.1.50.122
```

- Launch the Jupyter notebook from the server

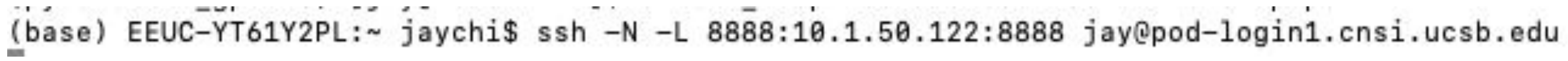
$ jupyter-notebook --no-browser --port=8888 --ip=10.1.50.122



```
    To access the server, open this file in a browser:
        file:///home/jay/.local/share/jupyter/runtime/jpserver-88004-open.html
    Or copy and paste one of these URLs:
        http://10.1.50.122:8888/tree?token=bff1d6512a520a13deb981d6c627d791198e25210536a840
        http://127.0.0.1:8888/tree?token=bff1d6512a520a13deb981d6c627d791198e25210536a840
[I 2024-02-26 15:29:41.318 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language
-server-nodejs, javascript-typescript-langserver, jedi-language-server, julia-language-server, pyright, python-l
anguage-server, python-lsp-server, r-languageserver, sql-language-server, texlab, typescript-language-server, un
ified-language-server, vscode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver
-bin, yaml-language-server
```

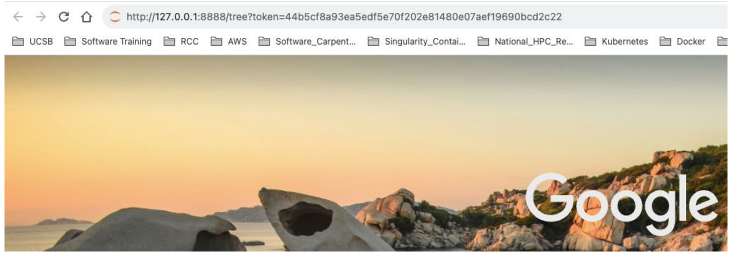
## Set Up Your Jupyter Notebook on the POD/Braid2

- Open a new terminal in order to access the Jupyter notebook from your remote machine over ssh
  $ ssh -N -L 8888:10.1.50.122:8888 your_user_name@pod-login1.cnsi.ucsb.edu



```
(base) EEUC-YT61Y2PL:~ jaychi$ ssh -N -L 8888:10.1.50.122:8888 jay@pod-login1.cnsi.ucsb.edu
```

- Open a browser window, copy the http://127.0.0.1:8888/tree?token=44b5cf8a93ea5edf5e70f202e81480e07aef19690bcd2c22 and pate it to the browser.



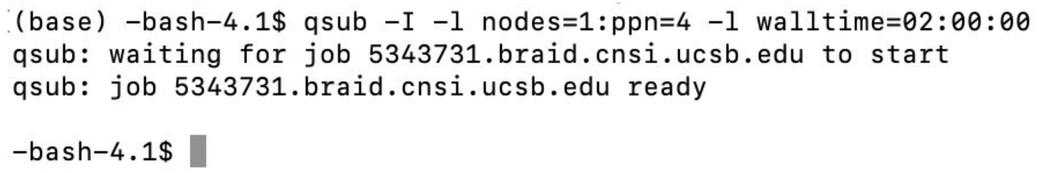
```
http://127.0.0.1:8888/tree?token=44b5cf8a93ea5edf5e70f202e81480e07aef19690bcd2c22
```

- After you finish your job, don't forget to release your resource.
  $ scancel your_job_id

## Set Up Your Jupyter Notebook on the Braid

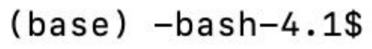
- Get to a compute node from the login node

```
$ qsub -I -l nodes=1:ppn=2 -l walltime=02:00:00
```



```
(base) -bash-4.1$ qsub -I -l nodes=1:ppn=4 -l walltime=02:00:00
qsub: waiting for job 5343731.braid.cnsi.ucsb.edu to start
qsub: job 5343731.braid.cnsi.ucsb.edu ready

-bash-4.1$
```

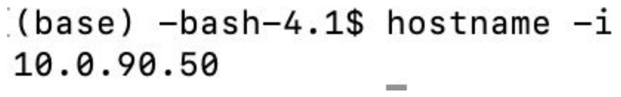
- Make sure your conda environment is activated



```
(base) -bash-4.1$
```

- Get the ip from the host

```
$ hostname -i
```

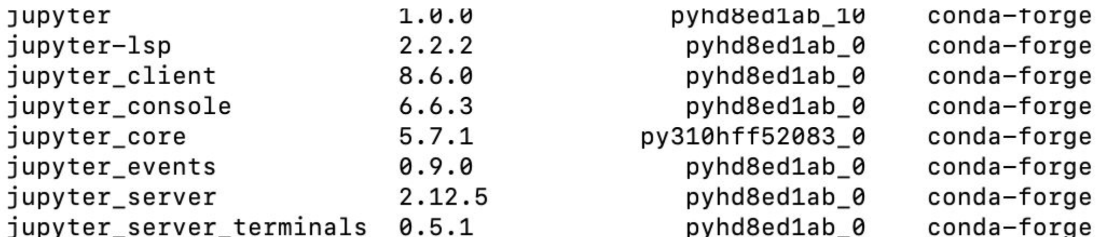


```
(base) -bash-4.1$ hostname -i
10.0.90.50
```

## Set Up Your Jupyter Notebook on the Braid

- Make sure the jupyter has been installed in the conda environment

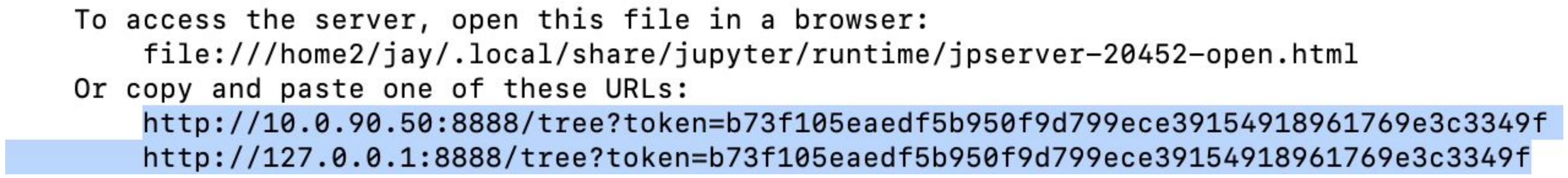
  $ conda list jupyter



```
jupyter                    1.0.0                pyhd8ed1ab_10    conda-forge
jupyter-lsp                2.2.2                 pyhd8ed1ab_0    conda-forge
jupyter_client             8.6.0                 pyhd8ed1ab_0    conda-forge
jupyter_console            6.6.3                 pyhd8ed1ab_0    conda-forge
jupyter_core               5.7.1              py310hff52083_0    conda-forge
jupyter_events             0.9.0                 pyhd8ed1ab_0    conda-forge
jupyter_server             2.12.5                pyhd8ed1ab_0    conda-forge
jupyter_server_terminals   0.5.1                 pyhd8ed1ab_0    conda-forge
```

- Launch the Jupyter notebook from the server

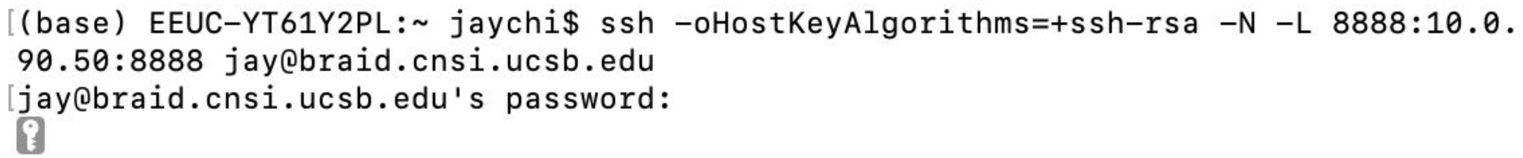
  $ jupyter notebook --no-browser --port=8888 --ip=10.0.90.50



```
To access the server, open this file in a browser:
    file:///home2/jay/.local/share/jupyter/runtime/jpserver-20452-open.html
Or copy and paste one of these URLs:
    http://10.0.90.50:8888/tree?token=b73f105eaedf5b950f9d799ece39154918961769e3c3349f
    http://127.0.0.1:8888/tree?token=b73f105eaedf5b950f9d799ece39154918961769e3c3349f
```

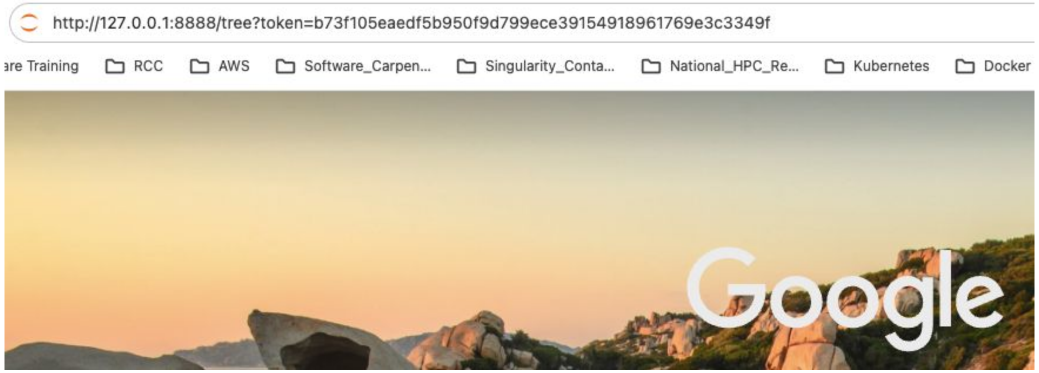
## Set Up Your Jupyter Notebook on the Braid

- Open a new terminal in order to access the Jupyter notebook from your local machine over ssh
  $ ssh -oHostKeyAlgorithms=+ssh-rsa -N -L 8888:10.0.90.50:8888 your_username@braid.cnsi.ucsb.edu



```
[(base) EEUC-YT61Y2PL:~ jaychi$ ssh -oHostKeyAlgorithms=+ssh-rsa -N -L 8888:10.0.
90.50:8888 jay@braid.cnsi.ucsb.edu
[jay@braid.cnsi.ucsb.edu's password:
```

- Open a browser window, copy the http://127.0.0.1:8888/tree?token=b73f105eaedf5b950f9d799ece39154918961769e3c3349f and pasta it to the browser.



http://127.0.0.1:8888/tree?token=b73f105eaedf5b950f9d799ece39154918961769e3c3349f

# Connect Visual Studio Code to POD

- According to the Wikipedia, "*Visual Studio Code (VS Code) is a source code editor that support a variety of programming languages, including C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust, and Julia.*"

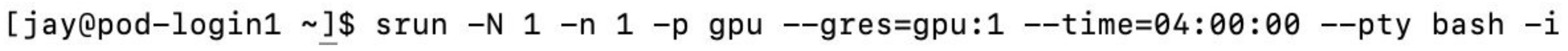
VS Code for

| | | | | | |
|---|---|---|---|---|---|
| JS JavaScript | Python | Java | Markdown | TS TypeScript | C/C++ |
| { } JSON | >_ Powershell | <> HTML/CSS | C# | PHP | YAML |

and many more languages on the Marketplace...

- GitHub Copilot is a code completion tool developed by GitHub and OpenAI that assists users of Visual Studio Code integrated development environments (IDEs) by autocompleting code.
- Get to a compute node from the login node

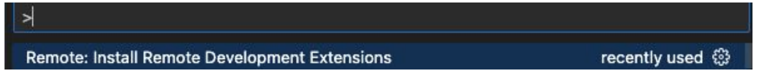
```
$ srun -N 1 -n 1 -p gpu --gres=gpu:1  --time=04:00:00 --pty bash -i
```

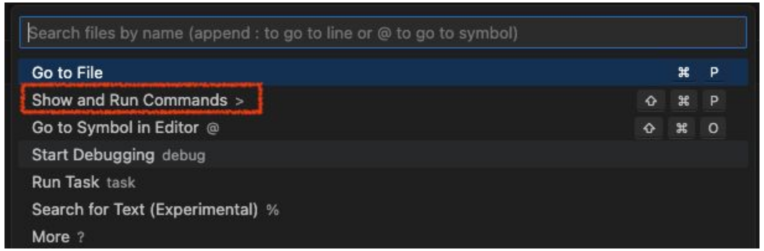


```
[jay@pod-login1 ~]$ srun -N 1 -n 1 -p gpu --gres=gpu:1 --time=04:00:00 --pty bash -i
```

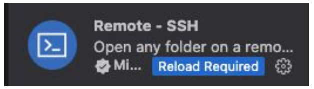
## Connect Visual Studio Code to POD

- Connect VS Code locally to the Computing Node in HPC
  - Open VS Code command palette

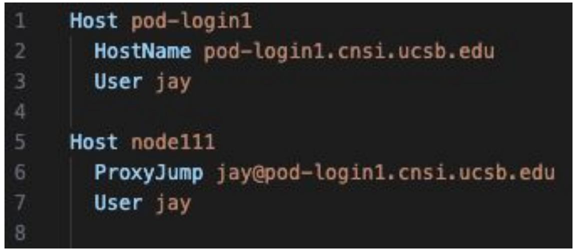

```
>|
Remote: Install Remote Development Extensions          recently used
```

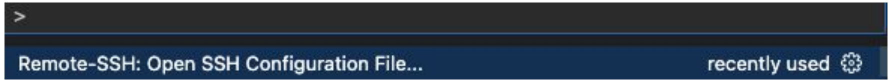

```
Search files by name (append : to go to line or @ to go to symbol)
Go to File                                   ⌘ P
Show and Run Commands >                      ⇧ ⌘ P
Go to Symbol in Editor @                     ⇧ ⌘ O
Start Debugging debug
Run Task task
Search for Text (Experimental) %
More ?
```

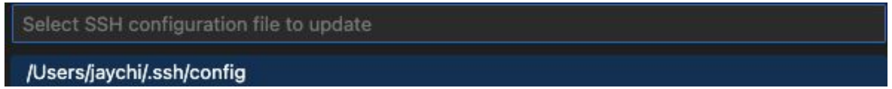
  - Install Remote - SSH



- Configuring SSH
  - Open SSH config file
  - Add the following config detail

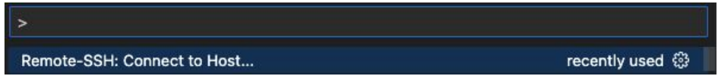

```
Host pod-login1
  HostName pod-login1.cnsi.ucsb.edu
  User jay

Host node111
  ProxyJump jay@pod-login1.cnsi.ucsb.edu
  User jay

```


```
>
Remote-SSH: Open SSH Configuration File...          recently used
```

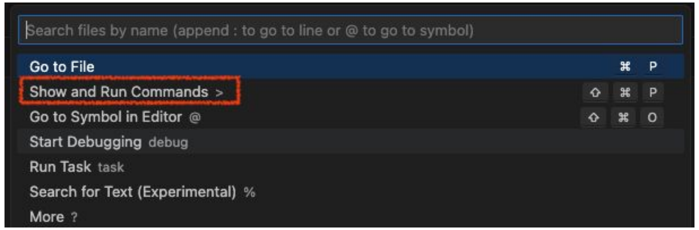

```
Select SSH configuration file to update
/Users/jaychi/.ssh/config
```

## Connect Visual Studio Code to POD

- Open VS Code command palette
  - Remote-SSH: Connect to Host

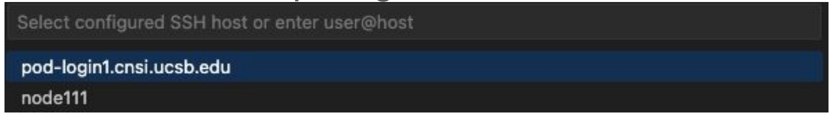


```
>
Remote-SSH: Connect to Host...          recently used
```



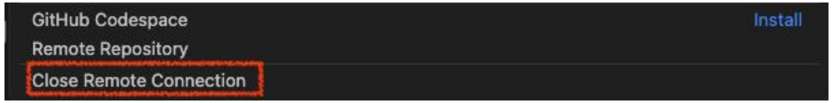
```
Search files by name (append : to go to line or @ to go to symbol)
Go to File                                   ⌘ P
Show and Run Commands >                    ⇧ ⌘ P
Go to Symbol in Editor @                   ⇧ ⌘ O
Start Debugging debug
Run Task task
Search for Text (Experimental) %
More ?
```

- Choose the computing node



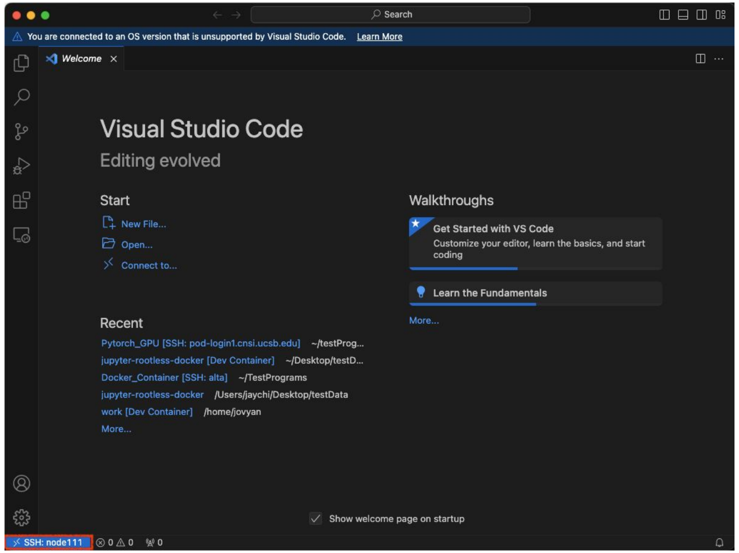
```
Select configured SSH host or enter user@host
pod-login1.cnsi.ucsb.edu
node111
```

- Disconnect to the HPC



```
GitHub Codespace                             Install
Remote Repository
Close Remote Connection
```

- After you finish your job, don't forget to release your resource.
  $ scancel your_job_id



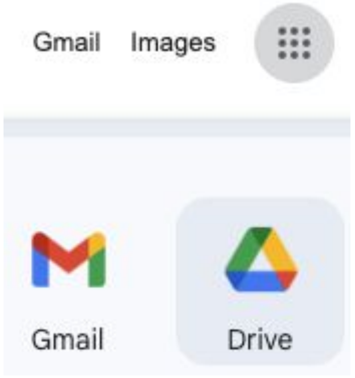
```
Search
You are connected to an OS version that is unsupported by Visual Studio Code. Learn More
Welcome ×

Visual Studio Code
Editing evolved

Start                          Walkthroughs
New File...                    Get Started with VS Code
Open...                        Customize your editor, learn the basics, and start coding
Connect to...                  Learn the Fundamentals
                               More...
Recent
Pytorch_GPU [SSH: pod-login1.cnsi.ucsb.edu]   ~/testProg...
jupyter-rootless-docker [Dev Container]   ~/Desktop/testD...
Docker_Container [SSH: alta]   ~/TestPrograms
jupyter-rootless-docker   /Users/jaychi/Desktop/testData
work [Dev Container]   /home/jovyan
More...

✓ Show welcome page on startup
SSH: node111   ⊗ 0 ⚠ 0   0
```

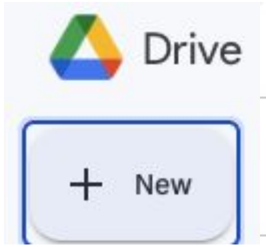
## Google Colab

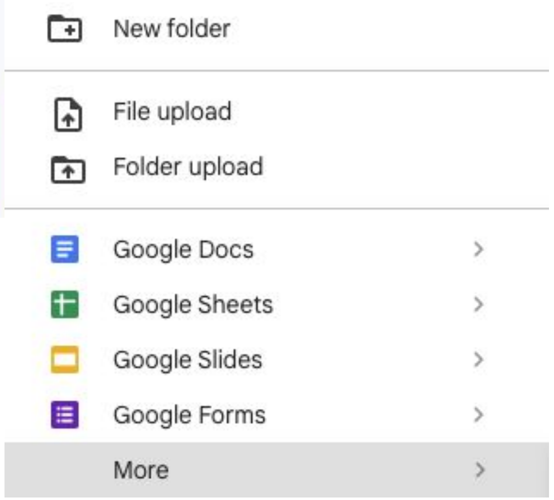
- What is Google Colab?
  - Google Colab(Colaboratory) allows you to write and execute Python and R in your browser with
    - No need to install packages
    - GPU access
    - Sharing with your partners

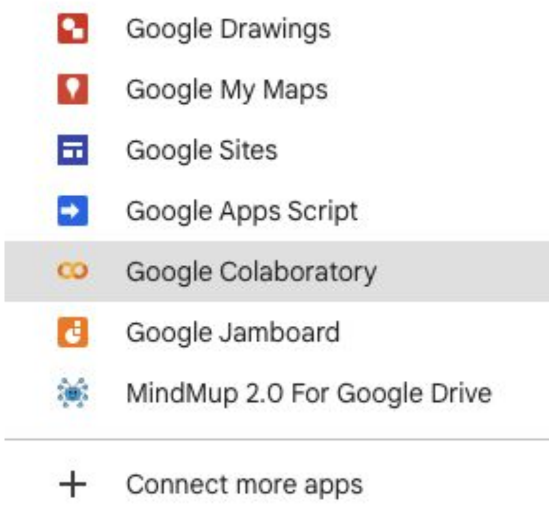
- Tutorial link
  - https://reurl.cc/Epg3M0

## Getting Started to use Colab

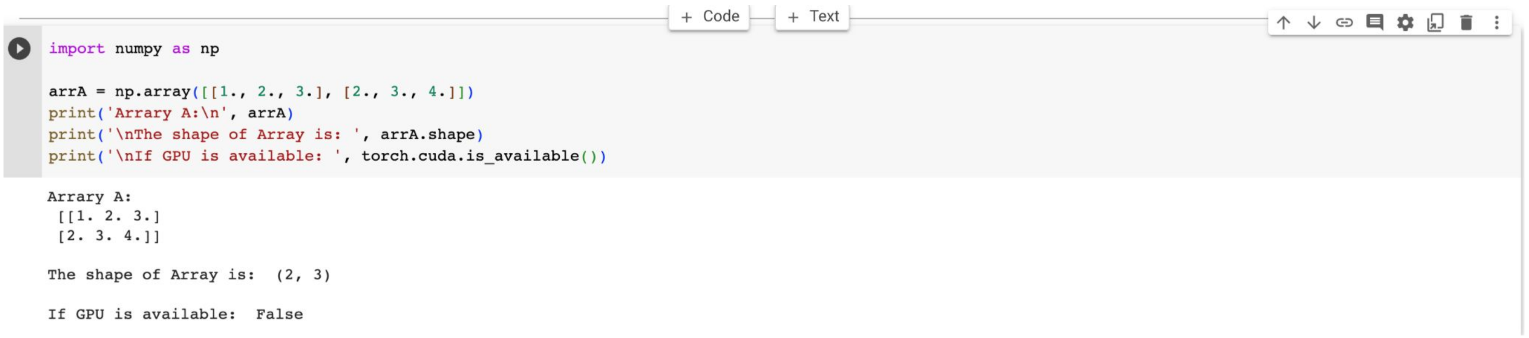








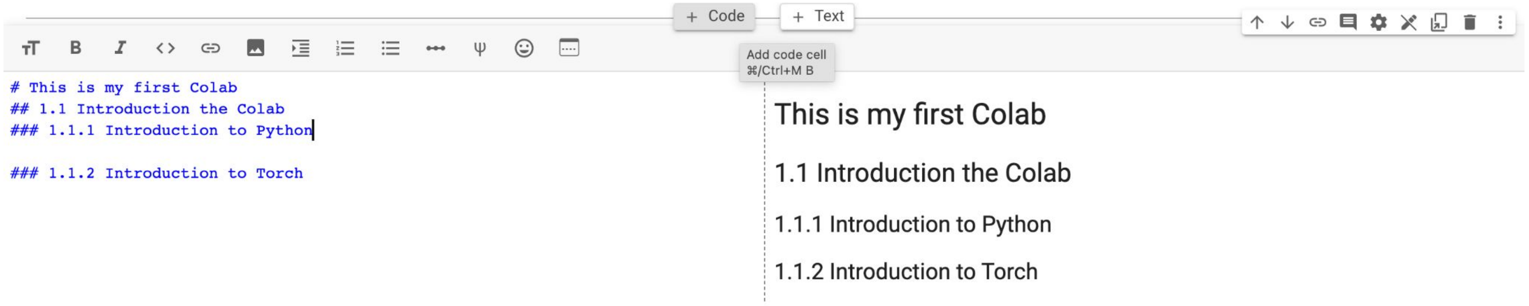
# Getting Started to use Colab



```
import numpy as np

arrA = np.array([[1., 2., 3.], [2., 3., 4.]])
print('Arrary A:\n', arrA)
print('\nThe shape of Array is: ', arrA.shape)
print('\nIf GPU is available: ', torch.cuda.is_available())
```

```
Arrary A:
 [[1. 2. 3.]
 [2. 3. 4.]]

The shape of Array is:  (2, 3)

If GPU is available:  False
```

Executing the Code Block: Shift + return



```
# This is my first Colab
## 1.1 Introduction the Colab
### 1.1.1 Introduction to Python

### 1.1.2 Introduction to Torch
```

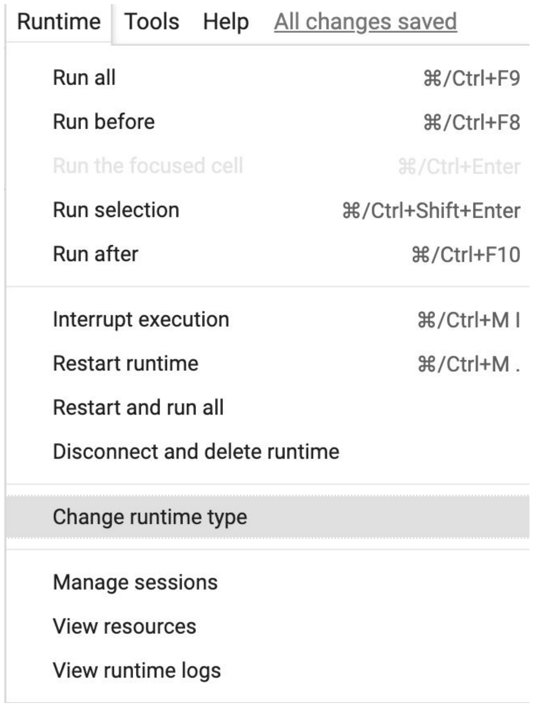
This is my first Colab

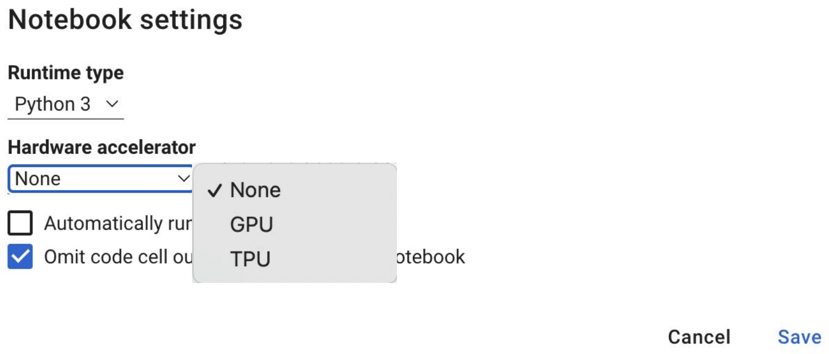
1.1 Introduction the Colab

1.1.1 Introduction to Python

1.1.2 Introduction to Torch

# Changing Runtime Type




**Notebook settings**

**Runtime type**
Python 3

**Hardware accelerator**
None

[✓ None] GPU TPU

☐ Automatically rur
☑ Omit code cell ou otebook

# ACCESS

Free National Supercomputer Resources

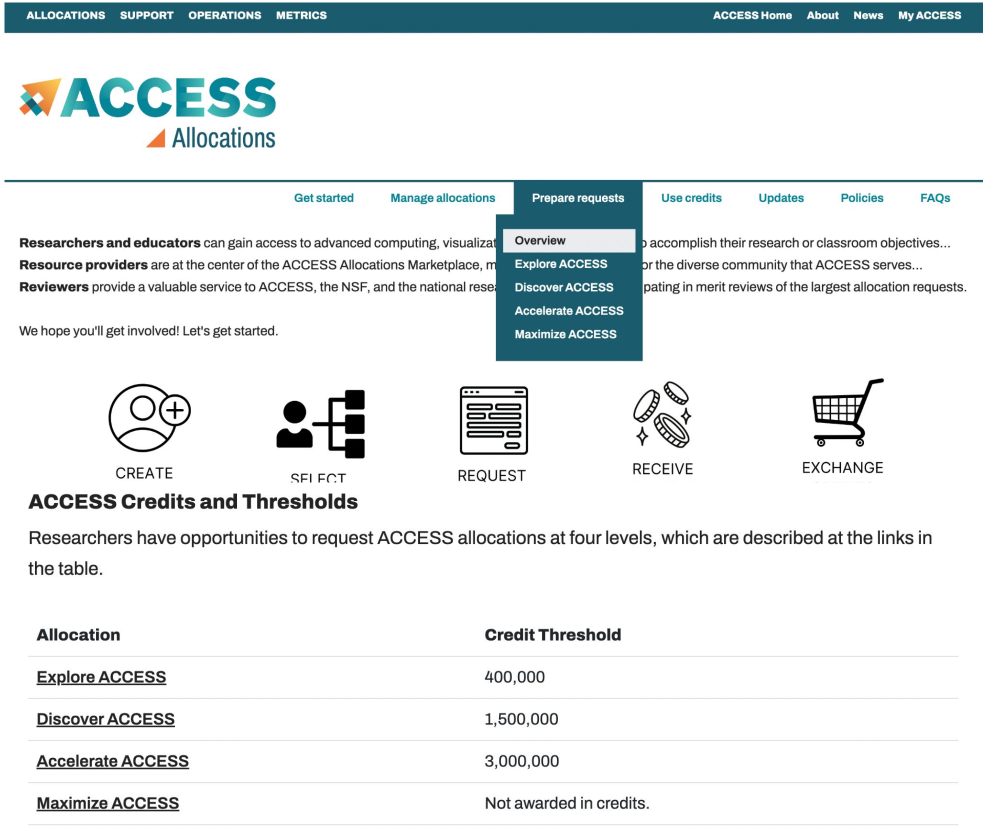
## National Supercomputer Resources: ACCESS

- Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS)
- ACCESS is an advanced computing and data resource supported by the National Science Foundation (NSF).
- ACCESS Services include Allocations, Support, Operation and Metrics, along with a Coordination Office
- Access website: https://access-ci.org/

# National Supercomputer Resources: ACCESS

**Four Allocation Opportunities** to suit a variety of needs (credit thresholds):

- ***Explore (400,000)***
  - Best-suited for endeavors with light resource requirements
    - Grad students can be PIs
- ***Discover (1,500,000)***
  - Minimal effort to start production research activities
    - NSF Graduate Fellowship
- ***Accelerate (3,000,000)***
  - More substantial resource requirements
    - Multi-grand research, Gateways, etc.
- ***Maximize (No upper limit)***
  - For large-scale research project with extreme resource needs
    - Will largely resemble XRAC process



**Researchers and educators** can gain access to advanced computing, visualizat o accomplish their research or classroom objectives...
**Resource providers** are at the center of the ACCESS Allocations Marketplace, m or the diverse community that ACCESS serves...
**Reviewers** provide a valuable service to ACCESS, the NSF, and the national resea pating in merit reviews of the largest allocation requests.

We hope you'll get involved! Let's get started.

**ACCESS Credits and Thresholds**

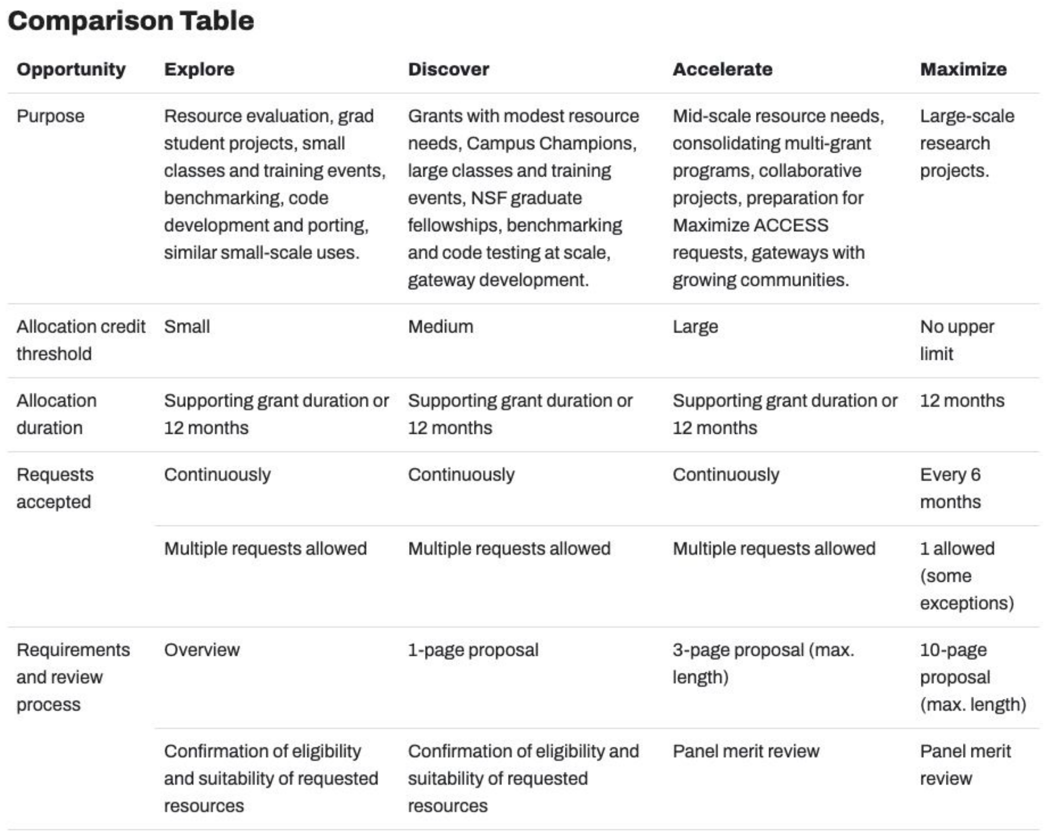
Researchers have opportunities to request ACCESS allocations at four levels, which are described at the links in the table.

| Allocation | Credit Threshold |
|---|---|
| Explore ACCESS | 400,000 |
| Discover ACCESS | 1,500,000 |
| Accelerate ACCESS | 3,000,000 |
| Maximize ACCESS | Not awarded in credits. |

## Allocation Eligibility

- Available to any research or educator as US academic, non-profit research, or educational institution.
- Can be in any official position including adjunct or instructional
- Postdoctoral researchers can be a PI of any project type
- Graduate students can lead an “Explore” ACCESS allocation under their advisor’s guidance
- NSF Graduate Fellows and Honorable mentions can apply for “Discover” allocations
- Ref: https://allocations.access-ci.org/access-allocations-policies#eligibility

## Comparison Table



### Comparison Table

| Opportunity | Explore | Discover | Accelerate | Maximize |
|---|---|---|---|---|
| Purpose | Resource evaluation, grad student projects, small classes and training events, benchmarking, code development and porting, similar small-scale uses. | Grants with modest resource needs, Campus Champions, large classes and training events, NSF graduate fellowships, benchmarking and code testing at scale, gateway development. | Mid-scale resource needs, consolidating multi-grant programs, collaborative projects, preparation for Maximize ACCESS requests, gateways with growing communities. | Large-scale research projects. |
| Allocation credit threshold | Small | Medium | Large | No upper limit |
| Allocation duration | Supporting grant duration or 12 months | Supporting grant duration or 12 months | Supporting grant duration or 12 months | 12 months |
| Requests accepted | Continuously | Continuously | Continuously | Every 6 months |
| | Multiple requests allowed | Multiple requests allowed | Multiple requests allowed | 1 allowed (some exceptions) |
| Requirements and review process | Overview | 1-page proposal | 3-page proposal (max. length) | 10-page proposal (max. length) |
| | Confirmation of eligibility and suitability of requested resources | Confirmation of eligibility and suitability of requested resources | Panel merit review | Panel merit review |

Ref: https://allocations.access-ci.org/prepare-requests-overview

## Resource Providers (RPs)

- ACCESS consists of a set of Resource Providers (PRs) that offer a wide range of computational resources including systems such as high-performance computing (HPC) clusters, virtualization (cloud-style) clusters, high throughput computing (HTC) clusters, massive storage clusters, large memory clusters, and composable clusters.

  - ACES (Texas A&M)
  - Anvil (Purdue)
  - Bridges-2 (PSC)
  - DARWIN (Delaware)
  - Delta (NCSA)
  - Expanse (SDSC)
  - FASTER (Texas A&M)
  - Jetstream2 (IU)
  - OOKAMI (Stonybrook)
  - KyRIC (Kentucky)
  - Rockfish (JHU)
  - Stampede-2 (TACC)
  - RANCH (TACC)
  - Open Science Grid (OSG)
  - Open Storage Network (OSN)

**Expanse**

San Diego Supercomputer Center | GPU Compute | ACCESS Pegasus | CPU Compute | ACCESS OnDemand | Large Memory Nodes | Advance reservations | Science Gateway support | Storage

Expanse is a dedicated ACCESS cluster designed by Dell and SDSC delivering 5.16 peak petaflops, and will offer Composable Systems and Cloud Bursting.

**Bridges-2**

Pittsburgh Supercomputing Center | GPU Compute | CPU Compute | Storage | Large Memory Nodes | Advance reservations | Science Gateway support | ACCESS OnDemand | ACCESS Pegasus

Bridges-2, a resource of Pittsburgh Supercomputing Center, is designed for converged HPC + AI + Data. Its custom topology is optimized for data-centric HPC, AI, and HPDA (High Performance Data Analytics). An extremely flexible software environment along with community data collections and BDaaS (Big Data as a Service) provide the tools necessary for modern pioneering research. The data management system, Ocean, consists of two-tiers, disk and tape, transparently managed as a single, highly usable namespace.

Jetstream2

**Jetstream2**

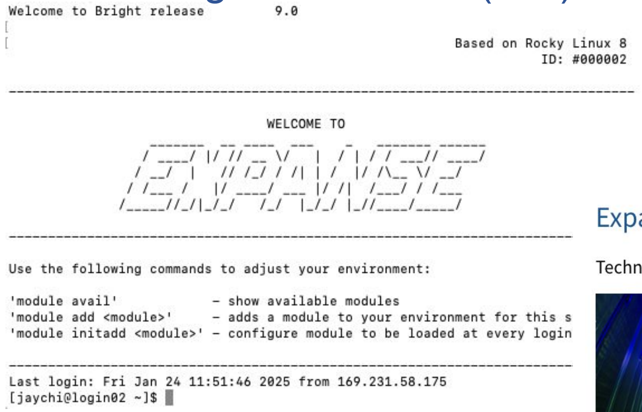
Indiana University | Cloud | GPU Compute | CPU Compute | Storage | Large Memory Nodes | Science Gateway support | ACCESS Pegasus

Jetstream2 is a transformative update to the NSF's science and engineering cloud infrastructure and provides 8 petaFLOPS of supercomputing power to simplify data analysis, boost discovery, and increase availability of AI resources. It is an NSF-funded, user-friendly cloud environment designed to allow "always on" research infrastructure and to give researchers access to interactive computing and data analysis resources on demand, whenever and wherever they want to analyze their data.

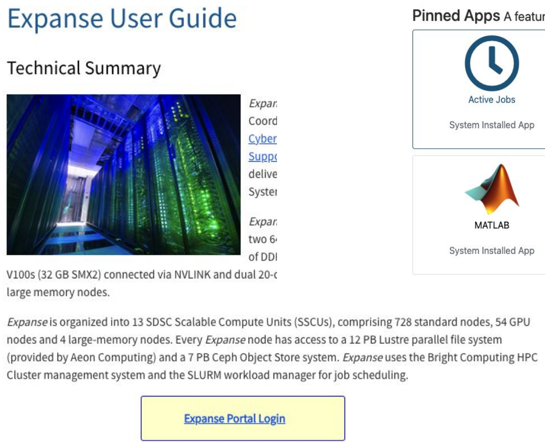
# Resource Provider: SDSC Expanse

- ssh to the SDSC Expanse HPC
  - ssh your_ACCESS_ID@login.expanse.sdsc.edu
- Connect to the Expanse Open OnDemand Portal: [https://portal.expanse.sdsc.edu/](https://portal.expanse.sdsc.edu/)

## Connecting via Terminal (CLI):



```
Welcome to Bright release         9.0

                                                        Based on Rocky Linux 8
                                                                ID: #000002

--------------------------------------------------------------------------------

                                WELCOME TO
                  ______  _______    ___    _   _______ ______
                 / ____/ |/ / __ \  /   |  / | / / ___// ____/
                / __/  |   / /_/ / / /| | /  |/ /\__ \/ __/
               / /___ /   / ____/ / ___ |/ /|  /___/ / /___
              /_____//_/|_/_/     /_/  |_/_/ |_//____/_____/

--------------------------------------------------------------------------------

Use the following commands to adjust your environment:

'module avail'            - show available modules
'module add <module>'     - adds a module to your environment for this s
'module initadd <module>' - configure module to be loaded at every login

--------------------------------------------------------------------------------
Last login: Fri Jan 24 11:51:46 2025 from 169.231.58.175
[jaychi@login02 ~]$ 
```



Expanse User Guide

Technical Summary

Expan
Coord
Cyber
Suppo
delive
Syster

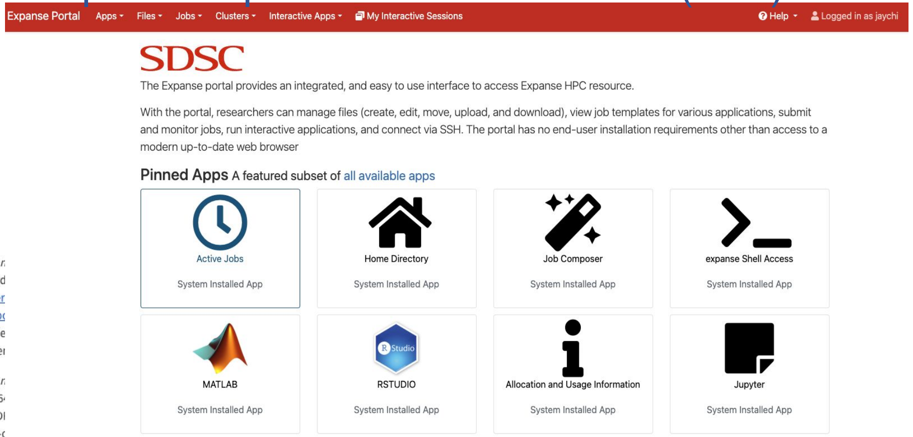
Expan
two 6
of DD

V100s (32 GB SMX2) connected via NVLINK and dual 20-c
large memory nodes.

*Expanse* is organized into 13 SDSC Scalable Compute Units (SSCUs), comprising 728 standard nodes, 54 GPU nodes and 4 large-memory nodes. Every *Expanse* node has access to a 12 PB Lustre parallel file system (provided by Aeon Computing) and a 7 PB Ceph Object Store system. *Expanse* uses the Bright Computing HPC Cluster management system and the SLURM workload manager for job scheduling.

Expanse Portal Login

## Expanse Open OnDemand Portal (GUI):



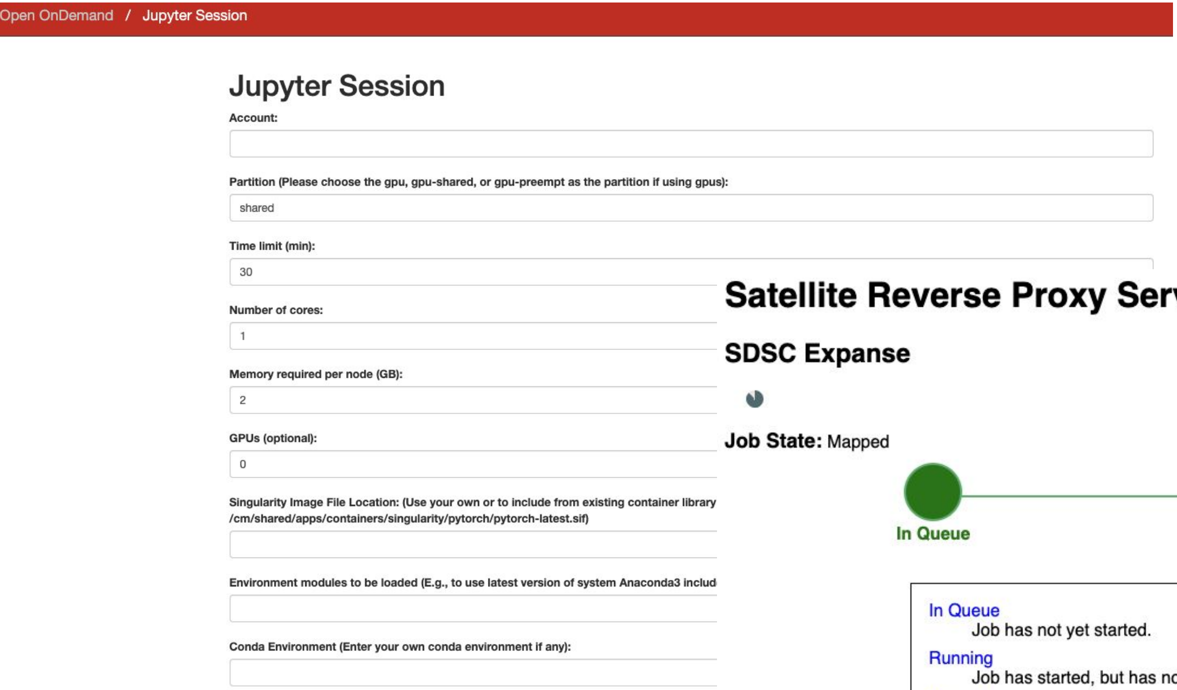
The Expanse portal provides an integrated, and easy to use interface to access Expanse HPC resource.

With the portal, researchers can manage files (create, edit, move, upload, and download), view job templates for various applications, submit and monitor jobs, run interactive applications, and connect via SSH. The portal has no end-user installation requirements other than access to a modern up-to-date web browser

**Pinned Apps** A featured subset of all available apps

Ref: [https://www.sdsc.edu/support/user_guides/expanse.html](https://www.sdsc.edu/support/user_guides/expanse.html)

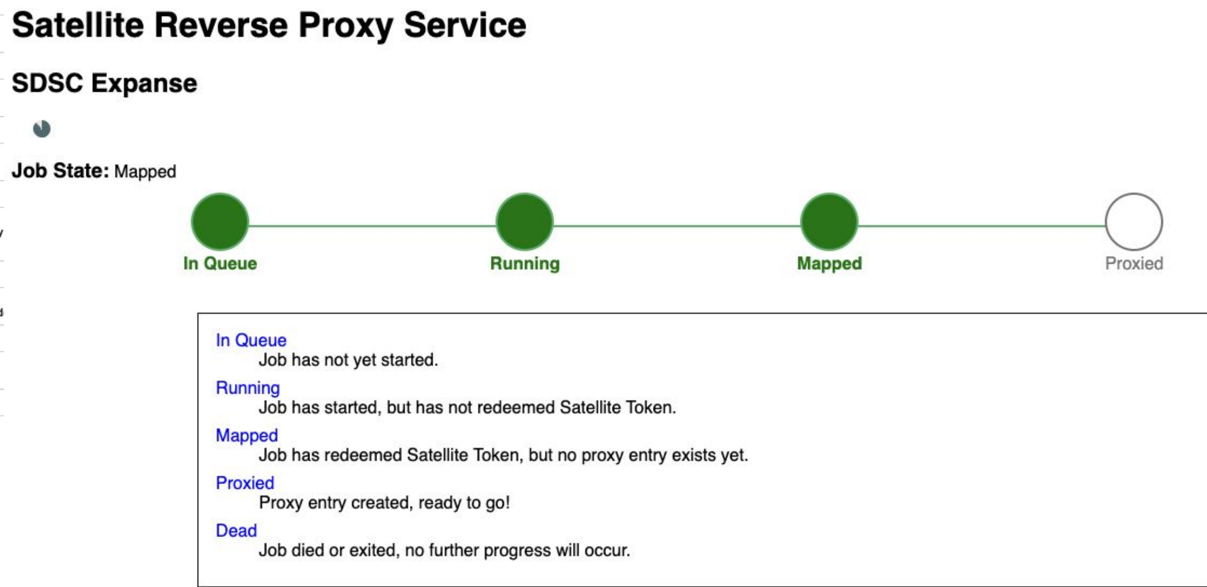
# Run Jupyter Notebook/Lab on the SDSC Expanse



Open OnDemand / Jupyter Session

## Jupyter Session

| | |
|---|---|
| Account: | |
| Partition (Please choose the gpu, gpu-shared, or gpu-preempt as the partition if using gpus): | shared |
| Time limit (min): | 30 |
| Number of cores: | 1 |
| Memory required per node (GB): | 2 |
| GPUs (optional): | 0 |
| Singularity Image File Location: (Use your own or to include from existing container library /cm/shared/apps/containers/singularity/pytorch/pytorch-latest.sif) | |
| Environment modules to be loaded (E.g., to use latest version of system Anaconda3 includ | |
| Conda Environment (Enter your own conda environment if any): | |



## Satellite Reverse Proxy Service

### SDSC Expanse

**Job State:** Mapped

In Queue
: Job has not yet started.

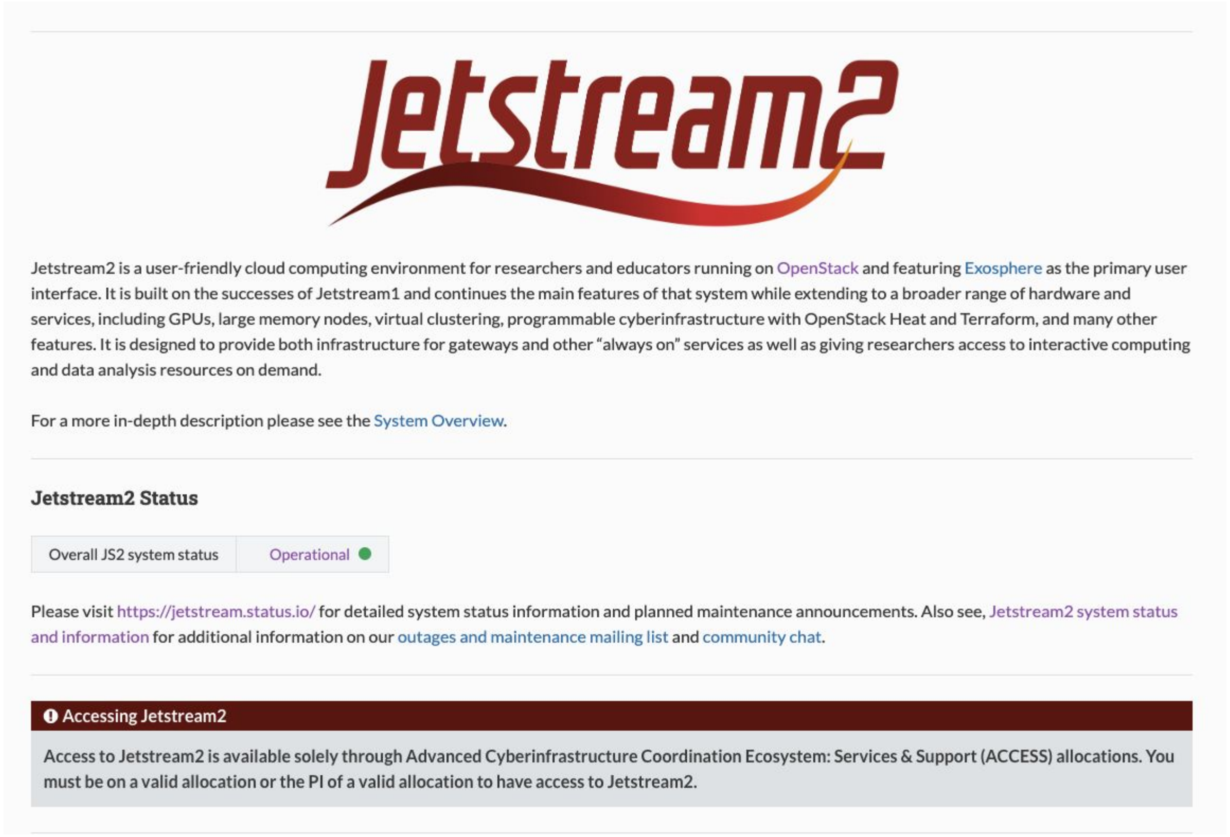
Running
: Job has started, but has not redeemed Satellite Token.

Mapped
: Job has redeemed Satellite Token, but no proxy entry exists yet.

Proxied
: Proxy entry created, ready to go!

Dead
: Job died or exited, no further progress will occur.

## RP: Indiana JstStream2



Jetstream2

Jetstream2 is a user-friendly cloud computing environment for researchers and educators running on OpenStack and featuring Exosphere as the primary user interface. It is built on the successes of Jetstream1 and continues the main features of that system while extending to a broader range of hardware and services, including GPUs, large memory nodes, virtual clustering, programmable cyberinfrastructure with OpenStack Heat and Terraform, and many other features. It is designed to provide both infrastructure for gateways and other "always on" services as well as giving researchers access to interactive computing and data analysis resources on demand.

For a more in-depth description please see the System Overview.

**Jetstream2 Status**

| | |
|---|---|
| Overall JS2 system status | Operational ● |

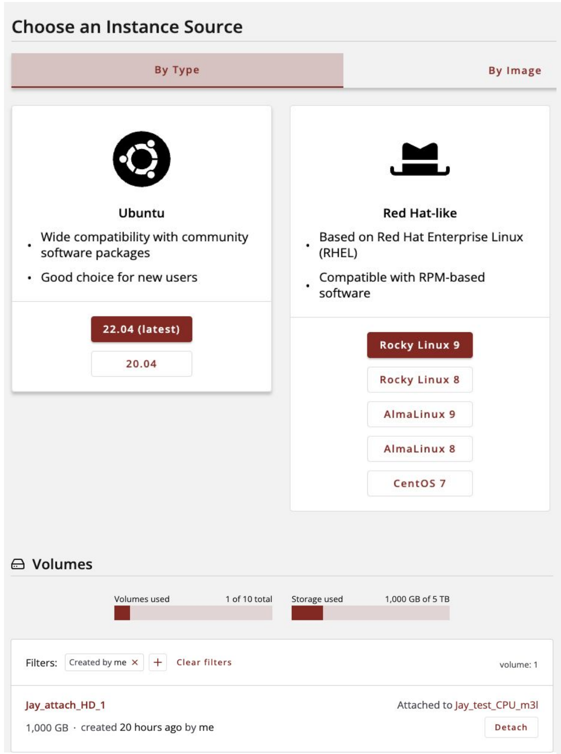
Please visit https://jetstream.status.io/ for detailed system status information and planned maintenance announcements. Also see, Jetstream2 system status and information for additional information on our outages and maintenance mailing list and community chat.

Accessing Jetstream2

Access to Jetstream2 is available solely through Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) allocations. You must be on a valid allocation or the PI of a valid allocation to have access to Jetstream2.

Ref: https://docs.jetstream-cloud.org/

# RP: Indiana JstStream2



## Choose an Instance Source

By Type | By Image

**Ubuntu**

- Wide compatibility with community software packages
- Good choice for new users

22.04 (latest) | 20.04

**Red Hat-like**

- Based on Red Hat Enterprise Linux (RHEL)
- Compatible with RPM-based software

Rocky Linux 9 | Rocky Linux 8 | AlmaLinux 9 | AlmaLinux 8 | CentOS 7

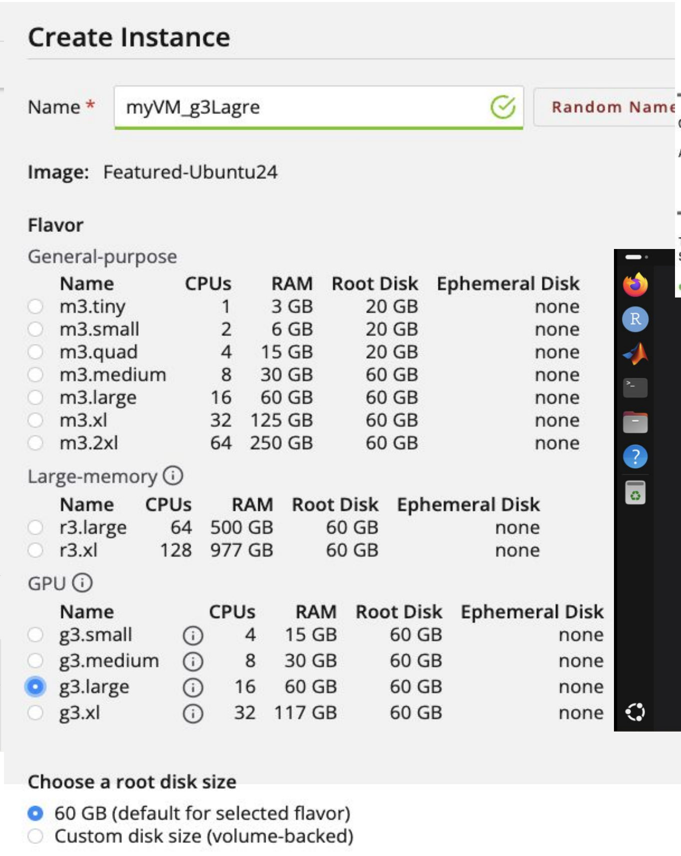
**Volumes**

| | |
|---|---|
| Volumes used | 1 of 10 total |
| Storage used | 1,000 GB of 5 TB |

Filters: Created by me × + Clear filters volume: 1

Jay_attach_HD_1 Attached to Jay_test_CPU_m3l

1,000 GB · created 20 hours ago by me Detach



## Create Instance

Name * myVM_g3Lagre Random Name

Image: Featured-Ubuntu24

**Flavor**

General-purpose

| Name | CPUs | RAM | Root Disk | Ephemeral Disk |
|---|---|---|---|---|
| m3.tiny | 1 | 3 GB | 20 GB | none |
| m3.small | 2 | 6 GB | 20 GB | none |
| m3.quad | 4 | 15 GB | 20 GB | none |
| m3.medium | 8 | 30 GB | 60 GB | none |
| m3.large | 16 | 60 GB | 60 GB | none |
| m3.xl | 32 | 125 GB | 60 GB | none |
| m3.2xl | 64 | 250 GB | 60 GB | none |

Large-memory

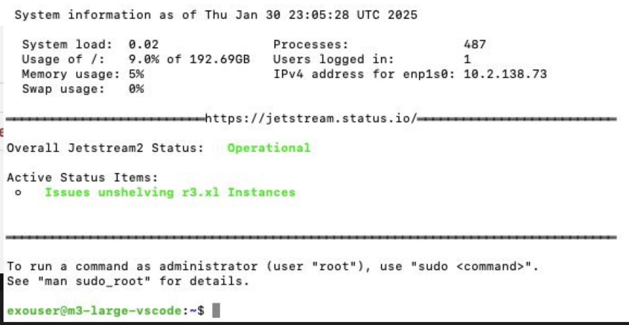
| Name | CPUs | RAM | Root Disk | Ephemeral Disk |
|---|---|---|---|---|
| r3.large | 64 | 500 GB | 60 GB | none |
| r3.xl | 128 | 977 GB | 60 GB | none |

GPU

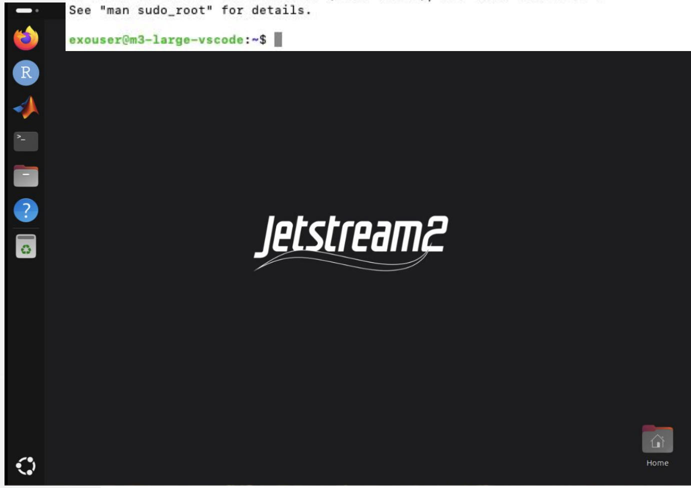
| Name | CPUs | RAM | Root Disk | Ephemeral Disk |
|---|---|---|---|---|
| g3.small | 4 | 15 GB | 60 GB | none |
| g3.medium | 8 | 30 GB | 60 GB | none |
| [✓ g3.large] | 16 | 60 GB | 60 GB | none |
| g3.xl | 32 | 117 GB | 60 GB | none |

**Choose a root disk size**

[✓ 60 GB (default for selected flavor)]
Custom disk size (volume-backed)

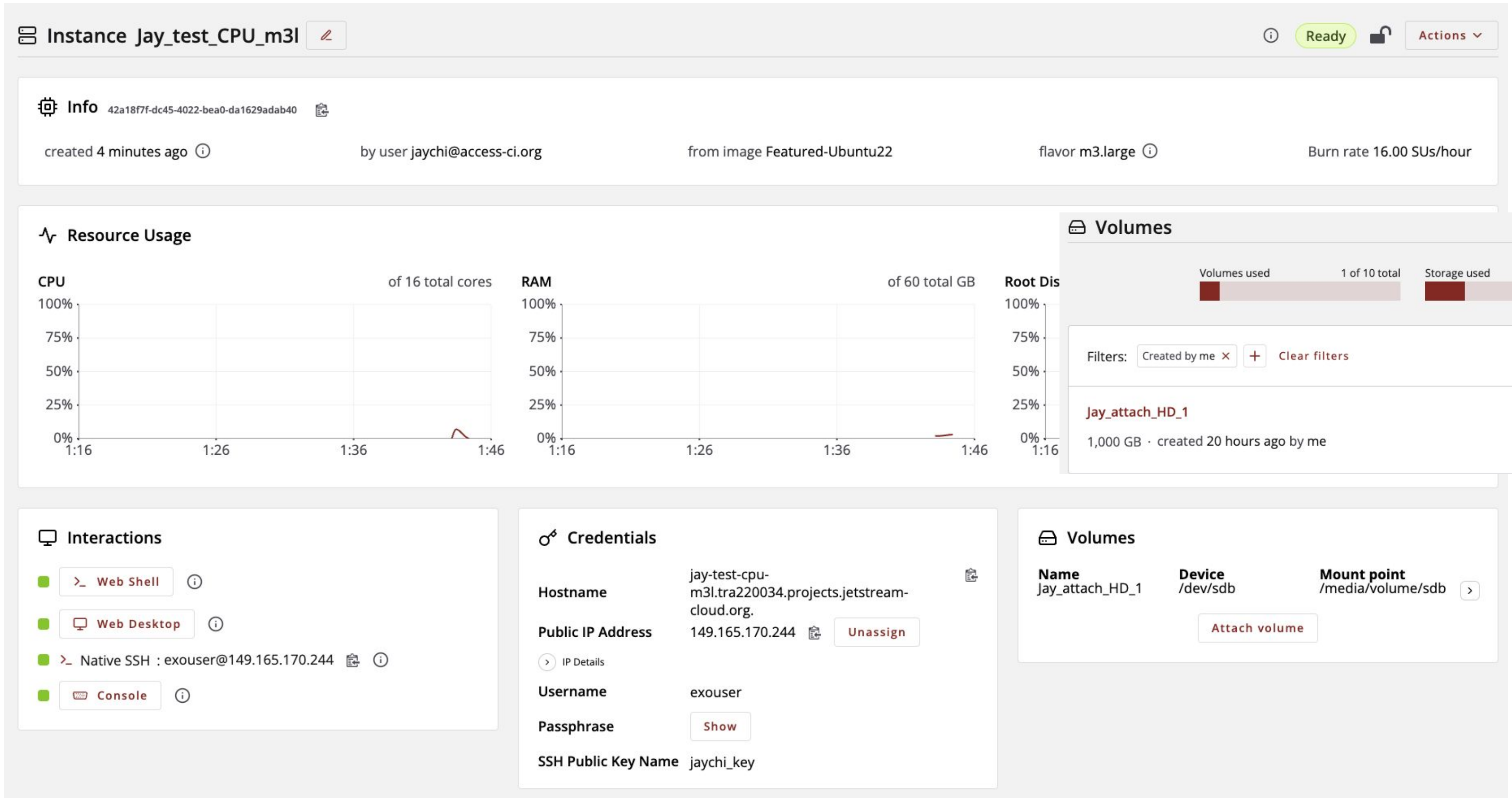


```
 System information as of Thu Jan 30 23:05:28 UTC 2025

  System load:  0.02               Processes:               487
  Usage of /:   9.0% of 192.69GB   Users logged in:         1
  Memory usage: 5%                 IPv4 address for enp1s0: 10.2.138.73
  Swap usage:   0%

━━━━━━━━━━━━━━━━━━━━━━━━━━https://jetstream.status.io/━━━━━━━━━━━━━━━━━━━━━━━━━━

Overall Jetstream2 Status:   Operational

Active Status Items:
 o   Issues unshelving r3.xl Instances


━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

exouser@m3-large-vscode:~$ 
```



https://jetstream2.exosphere.app/exosphere/

# RP: Indiana JstStream2



Instance Jay_test_CPU_m3l

Ready

Actions

Info 42a18f7f-dc45-4022-bea0-da1629adab40

created 4 minutes ago · by user jaychi@access-ci.org · from image Featured-Ubuntu22 · flavor m3.large · Burn rate 16.00 SUs/hour

Resource Usage

Interactions

- Web Shell
- Web Desktop
- Native SSH : exouser@149.165.170.244
- Console

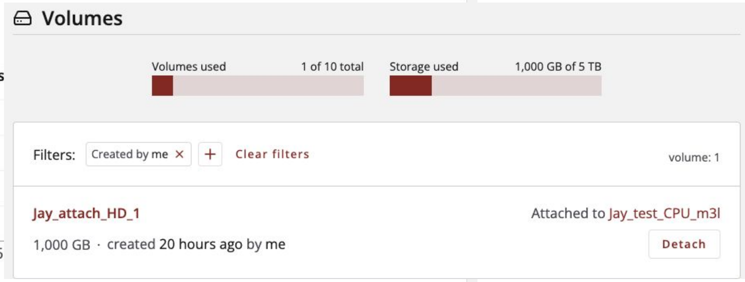
Credentials

| | |
|---|---|
| Hostname | jay-test-cpu-m3l.tra220034.projects.jetstream-cloud.org. |
| Public IP Address | 149.165.170.244 Unassign |
| IP Details | |
| Username | exouser |
| Passphrase | Show |
| SSH Public Key Name | jaychi_key |

Volumes

| Name | Device | Mount point |
|---|---|---|
| Jay_attach_HD_1 | /dev/sdb | /media/volume/sdb |

Attach volume



Volumes

Volumes used 1 of 10 total

Storage used 1,000 GB of 5 TB

Filters: Created by me × + Clear filters volume: 1

Jay_attach_HD_1 Attached to Jay_test_CPU_m3l

1,000 GB · created 20 hours ago by me

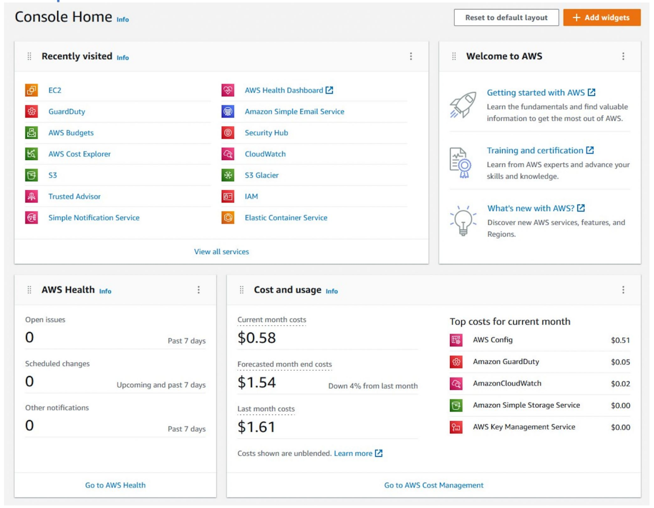
Detach

# Cloud Computing: Amazon Web Services (AWS)

- If you choose to use AWS, it is recommended to take advantage of the Campus Cloud Landing Zone (LZ) for AWS. A UCSB purchases order is required to request an Campus Cloud account (https://ucsb.github.io/campus-cloud-docs/getting-started/#procurement).

Campus Single Sign On for AWS: https://aws.cloud.ucsb.edu

- Supported Campus Cloud Regions:
  - **US-West-2 (Oregon)**
  - **US-East-1 (N.Virginia)**
- We recommend starting in *US-West-2*



Console Home Info

Reset to default layout | + Add widgets

**Recently visited** Info

| | |
|---|---|
| EC2 | AWS Health Dashboard |
| GuardDuty | Amazon Simple Email Service |
| AWS Budgets | Security Hub |
| AWS Cost Explorer | CloudWatch |
| S3 | S3 Glacier |
| Trusted Advisor | IAM |
| Simple Notification Service | Elastic Container Service |

View all services

**Welcome to AWS**

Getting started with AWS
Learn the fundamentals and find valuable information to get the most out of AWS.

Training and certification
Learn from AWS experts and advance your skills and knowledge.

What's new with AWS?
Discover new AWS services, features, and Regions.

**AWS Health** Info

Open issues
0 Past 7 days

Scheduled changes
0 Upcoming and past 7 days

Other notifications
0 Past 7 days

Go to AWS Health

**Cost and usage** Info

Current month costs
$0.58

Forecasted month end costs
$1.54 Down 4% from last month

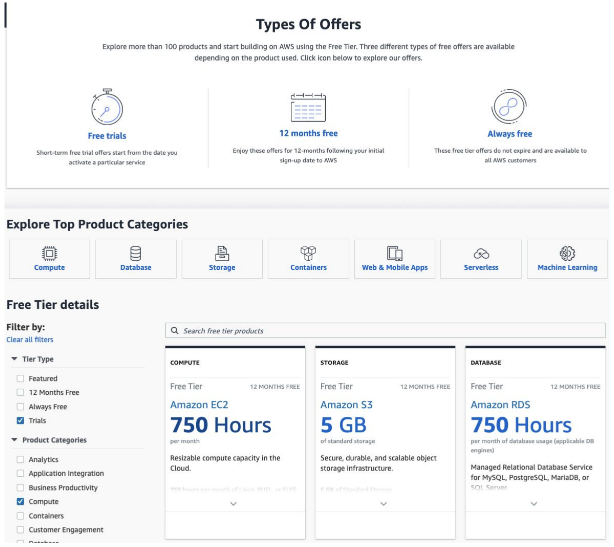
Last month costs
$1.61

Costs shown are unblended. Learn more

Top costs for current month

| | |
|---|---|
| AWS Config | $0.51 |
| Amazon GuardDuty | $0.05 |
| AmazonCloudWatch | $0.02 |
| Amazon Simple Storage Service | $0.00 |
| AWS Key Management Service | $0.00 |

Go to AWS Cost Management



**Types Of Offers**

Explore more than 100 products and start building on AWS using the Free Tier. Three different types of free offers are available depending on the product used. Click icon below to explore our offers.

Free trials
Short-term free trial offers start from the date you activate a particular service

12 months free
Enjoy these offers for 12-months following your initial sign-up date to AWS

Always free
These free tier offers do not expire and are available to all AWS customers

**Explore Top Product Categories**

Compute | Database | Storage | Containers | Web & Mobile Apps | Serverless | Machine Learning

**Free Tier details**

Filter by:
Clear all filters

Tier Type
- [ ] Featured
- [ ] 12 Months Free
- [ ] Always Free
- [✓] Trials

Product Categories
- [ ] Analytics
- [ ] Application Integration
- [ ] Business Productivity
- [✓] Compute
- [ ] Containers
- [ ] Customer Engagement

Search free tier products

COMPUTE
Free Tier 12 MONTHS FREE
Amazon EC2
750 Hours
per month
Resizable compute capacity in the Cloud.
750 hours per month of Linux, RHEL, or SLES

STORAGE
Free Tier 12 MONTHS FREE
Amazon S3
5 GB
of standard storage
Secure, durable, and scalable object storage infrastructure.
5 GB of Standard Storage

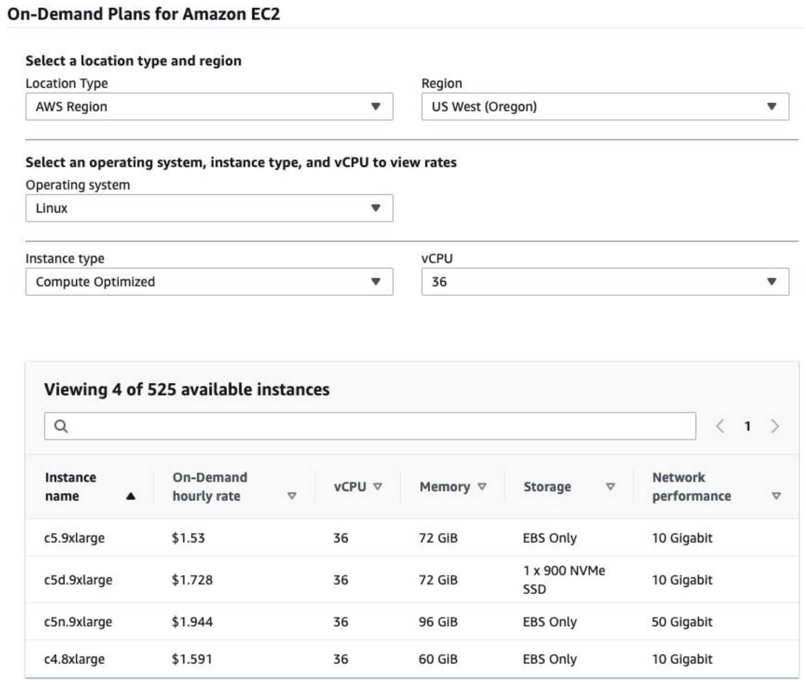
DATABASE
Free Tier 12 MONTHS FREE
Amazon RDS
750 Hours
per month of database usage (applicable DB engines)
Managed Relational Database Service for MySQL, PostgreSQL, MariaDB, or SQL Server.

***Important:*** You may need the help of a PI or Department Purchasing person to create a Purchase Order which is necessary to request an account in the Campus Cloud.

# Amazon Elastic Compute Cloud (Amazon EC2)

- Use Case:
  - Run cloud-native and enterprise applications
  - Scale for HPC applications
  - Train and deploy ML applications
- EC2 Instance Types
  - General Purpose
  - Compute Optimized
  - Memory Optimization
  - Accelerated Computing
  - Storage Optimized
- More Information
  - Amazon EC2: https://aws.amazon.com/ec2/
  - Amazon EC2 Pricing Estimation: https://aws.amazon.com/ec2/pricing/on-demand/

    https://instances.vantage.sh/



**On-Demand Plans for Amazon EC2**

**Select a location type and region**

Location Type: AWS Region

Region: US West (Oregon)

**Select an operating system, instance type, and vCPU to view rates**

Operating system: Linux

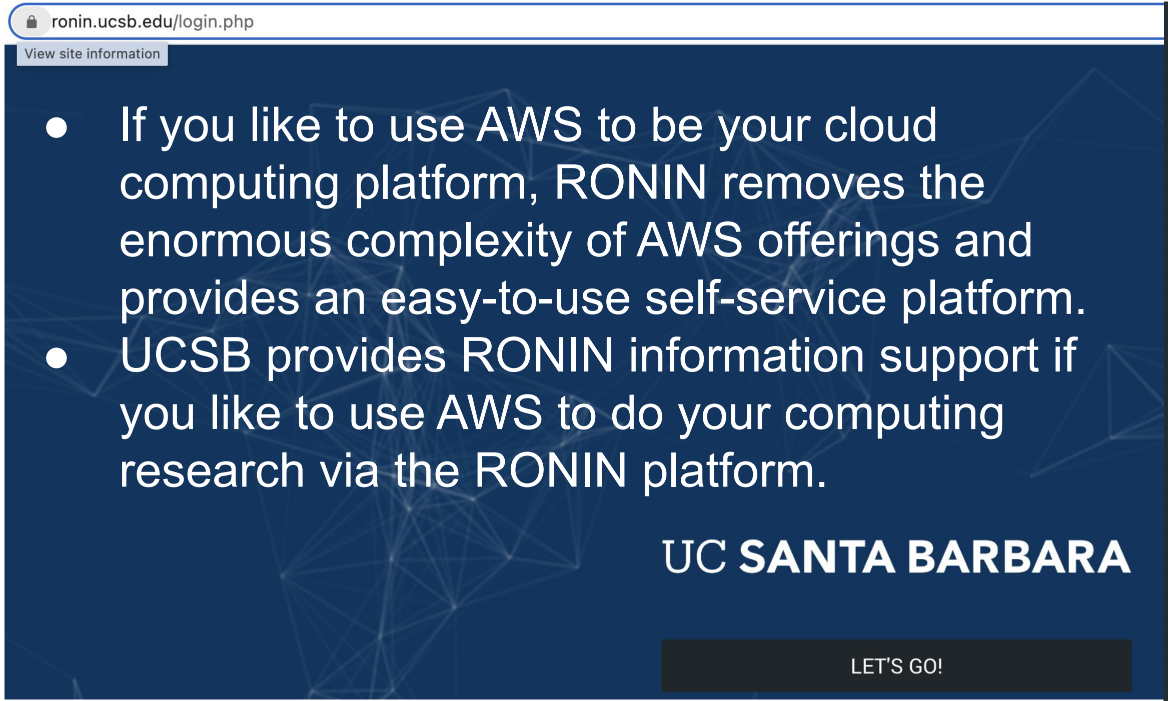
Instance type: Compute Optimized

vCPU: 36

**Viewing 4 of 525 available instances**

| Instance name | On-Demand hourly rate | vCPU | Memory | Storage | Network performance |
|---|---|---|---|---|---|
| c5.9xlarge | $1.53 | 36 | 72 GiB | EBS Only | 10 Gigabit |
| c5d.9xlarge | $1.728 | 36 | 72 GiB | 1 x 900 NVMe SSD | 10 Gigabit |
| c5n.9xlarge | $1.944 | 36 | 96 GiB | EBS Only | 50 Gigabit |
| c4.8xlarge | $1.591 | 36 | 60 GiB | EBS Only | 10 Gigabit |

## Ronin Platform RONIN

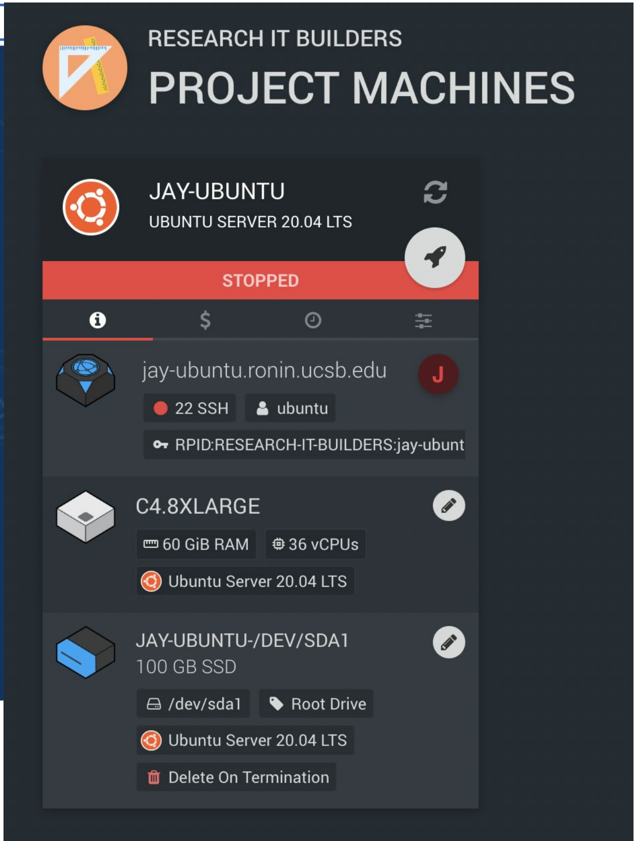

ronin.ucsb.edu/login.php

View site information

- If you like to use AWS to be your cloud computing platform, RONIN removes the enormous complexity of AWS offerings and provides an easy-to-use self-service platform.
- UCSB provides RONIN information support if you like to use AWS to do your computing research via the RONIN platform.

UC SANTA BARBARA

LET'S GO!


RESEARCH IT BUILDERS

PROJECT MACHINES

JAY-UBUNTU

UBUNTU SERVER 20.04 LTS

STOPPED

jay-ubuntu.ronin.ucsb.edu

22 SSH | ubuntu

RPID:RESEARCH-IT-BUILDERS:jay-ubunt

C4.8XLARGE

60 GiB RAM | 36 vCPUs

Ubuntu Server 20.04 LTS

JAY-UBUNTU-/DEV/SDA1

100 GB SSD

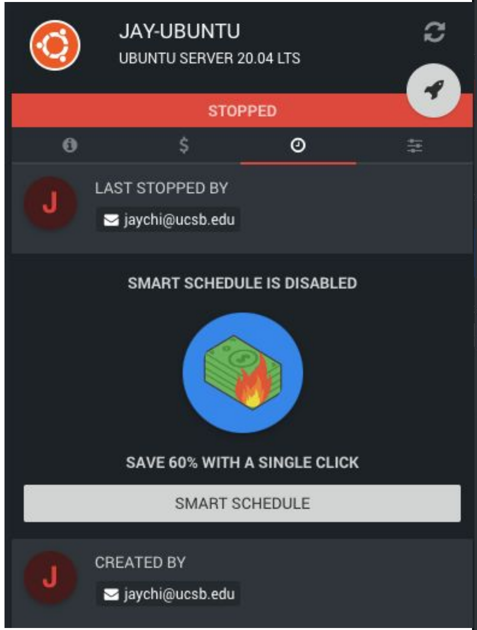
/dev/sda1 | Root Drive

Ubuntu Server 20.04 LTS

Delete On Termination

Contact with Bill Doering: billd@ucsb.edu

# Ronin Platform: Control Your AWS Cost



JAY-UBUNTU
UBUNTU SERVER 20.04 LTS

STOPPED

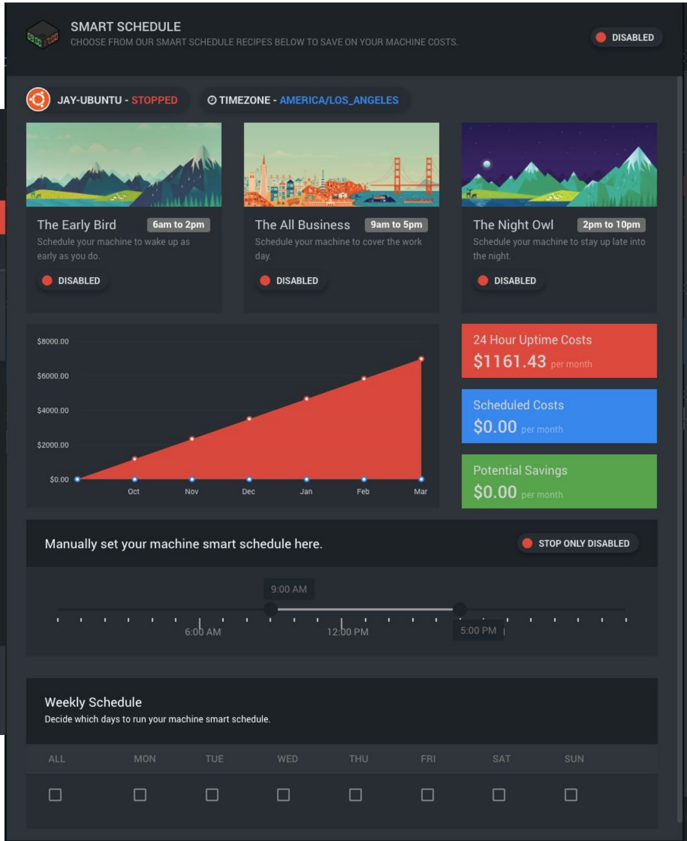
LAST STOPPED BY
jaychi@ucsb.edu

SMART SCHEDULE IS DISABLED

SAVE 60% WITH A SINGLE CLICK

SMART SCHEDULE

CREATED BY
jaychi@ucsb.edu



SMART SCHEDULE
CHOOSE FROM OUR SMART SCHEDULE RECIPES BELOW TO SAVE ON YOUR MACHINE COSTS.
DISABLED

JAY-UBUNTU - STOPPED  TIMEZONE - AMERICA/LOS_ANGELES

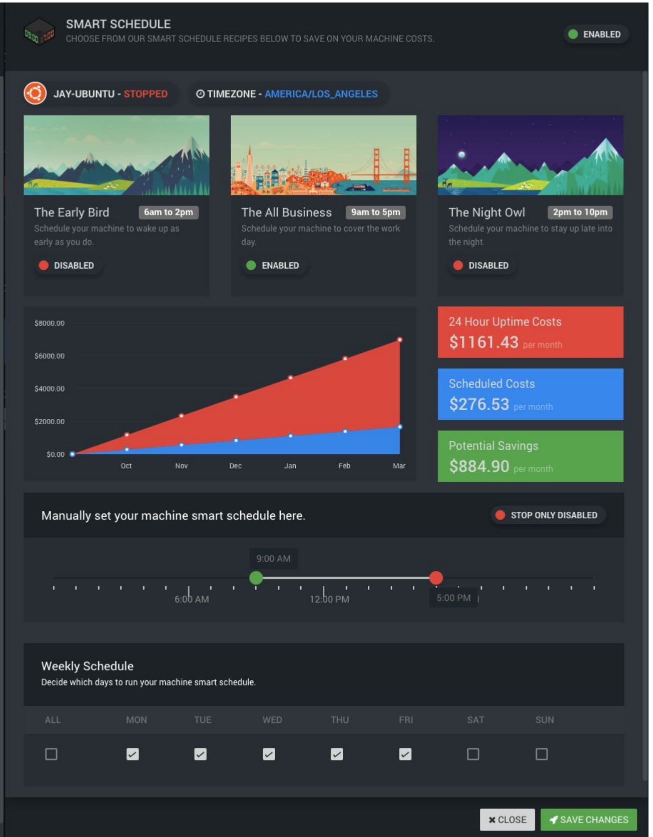
| The Early Bird 6am to 2pm | The All Business 9am to 5pm | The Night Owl 2pm to 10pm |
|---|---|---|
| Schedule your machine to wake up as early as you do. | Schedule your machine to cover the work day. | Schedule your machine to stay up late into the night. |
| DISABLED | DISABLED | DISABLED |

24 Hour Uptime Costs $1161.43 per month
Scheduled Costs $0.00 per month
Potential Savings $0.00 per month

Manually set your machine smart schedule here. STOP ONLY DISABLED

Weekly Schedule
Decide which days to run your machine smart schedule.

| ALL | MON | TUE | WED | THU | FRI | SAT | SUN |
|---|---|---|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ |



SMART SCHEDULE
CHOOSE FROM OUR SMART SCHEDULE RECIPES BELOW TO SAVE ON YOUR MACHINE COSTS.
ENABLED

JAY-UBUNTU - STOPPED  TIMEZONE - AMERICA/LOS_ANGELES

| The Early Bird 6am to 2pm | The All Business 9am to 5pm | The Night Owl 2pm to 10pm |
|---|---|---|
| Schedule your machine to wake up as early as you do. | Schedule your machine to cover the work day. | Schedule your machine to stay up late into the night. |
| DISABLED | ENABLED | DISABLED |

24 Hour Uptime Costs $1161.43 per month
Scheduled Costs $276.53 per month
Potential Savings $884.90 per month

Manually set your machine smart schedule here. STOP ONLY DISABLED

Weekly Schedule
Decide which days to run your machine smart schedule.

| ALL | MON | TUE | WED | THU | FRI | SAT | SUN |
|---|---|---|---|---|---|---|---|
| ☐ | ✓ | ✓ | ✓ | ✓ | ✓ | ☐ | ☐ |

## Ack!

- Acknowledgements - [https://csc.cnsi.ucsb.edu/publications](https://csc.cnsi.ucsb.edu/publications)

Please acknowledge the CSC in publications and presentations if you are using our facility's computational resources (including staff involvement) in your research.

"We acknowledge support from the Center for Scientific Computing from the CNSI, MRL: an NSF MRSEC (DMR-2308708) and NSF CNS- 1725797."

For users of GPU nodes, please add the grant number NSF OAC-1925717

## Questions and Thought

- What else content should we cover?
- Other ideas for a workshop?
    - Running Parallel Python / Matlab / R on the Cluster, Mathematica, Singularity/Docker Container, etc.

- More Information:

  https://csc.cnsi.ucsb.edu/