HIGH PERFORMANCE RESEARCH COMPUTING

ACES: Advanced Python for High Performance Computing Workflows

December 2, 2025

Richard Lawrence

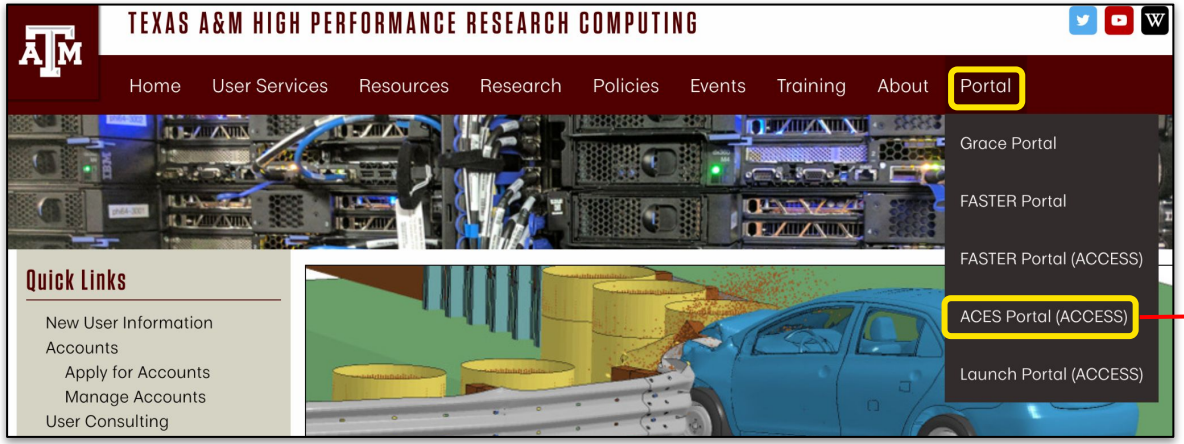
High Performance
Research Computing
DIVISION OF RESEARCH

Outline

- ACES Login
- Modules
- Jupyter
- Virtual Environments
- Conda Environments
- Containers
- Parallelism with Joblib and Dask
- Checkpointing with Pickle

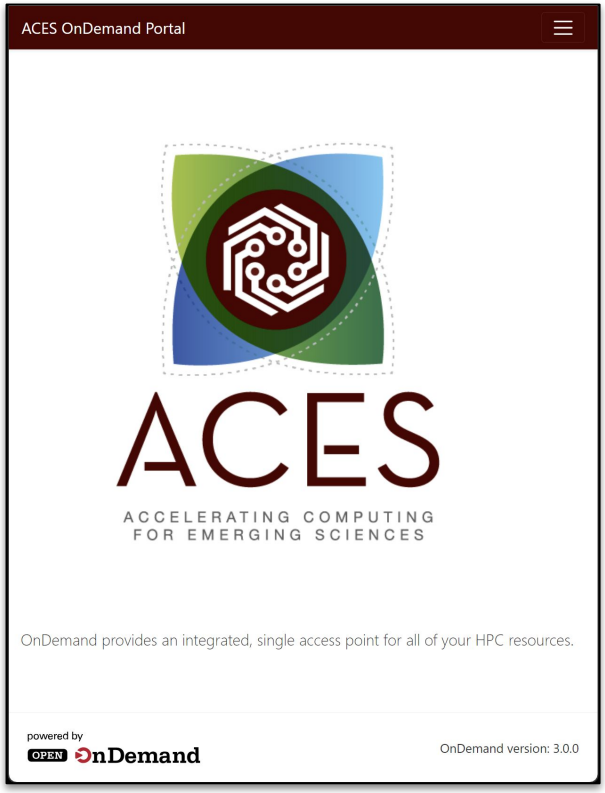
ACES Login

ACES Portal

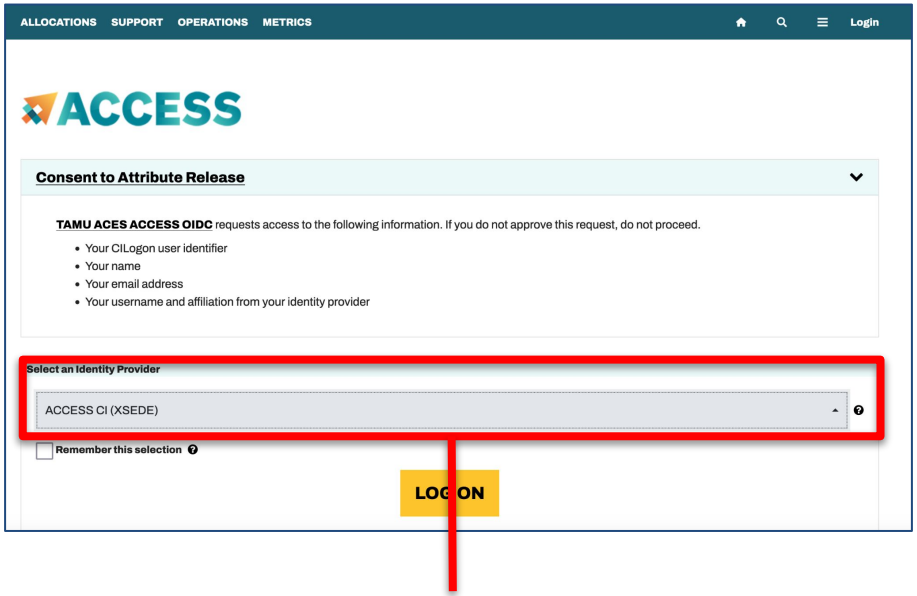


ACES Portal portal-aces.hprc.tamu.edu
is the web-based user interface for the ACES cluster

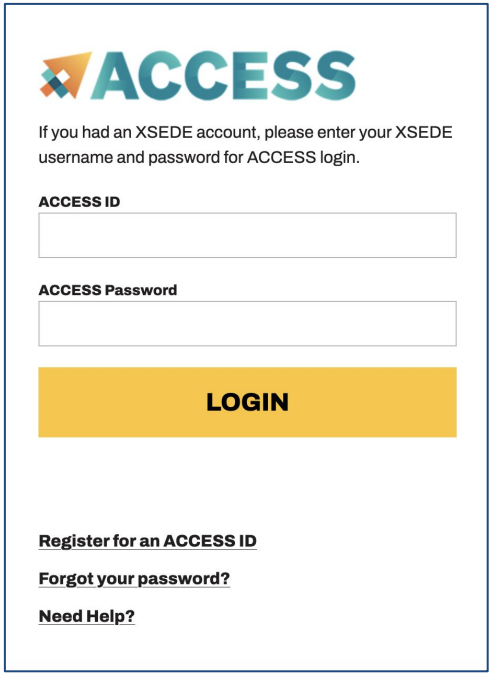
Open OnDemand (OOD) is an advanced web-based graphical interface framework for HPC users



Accessing ACES via the Portal (ACCESS)

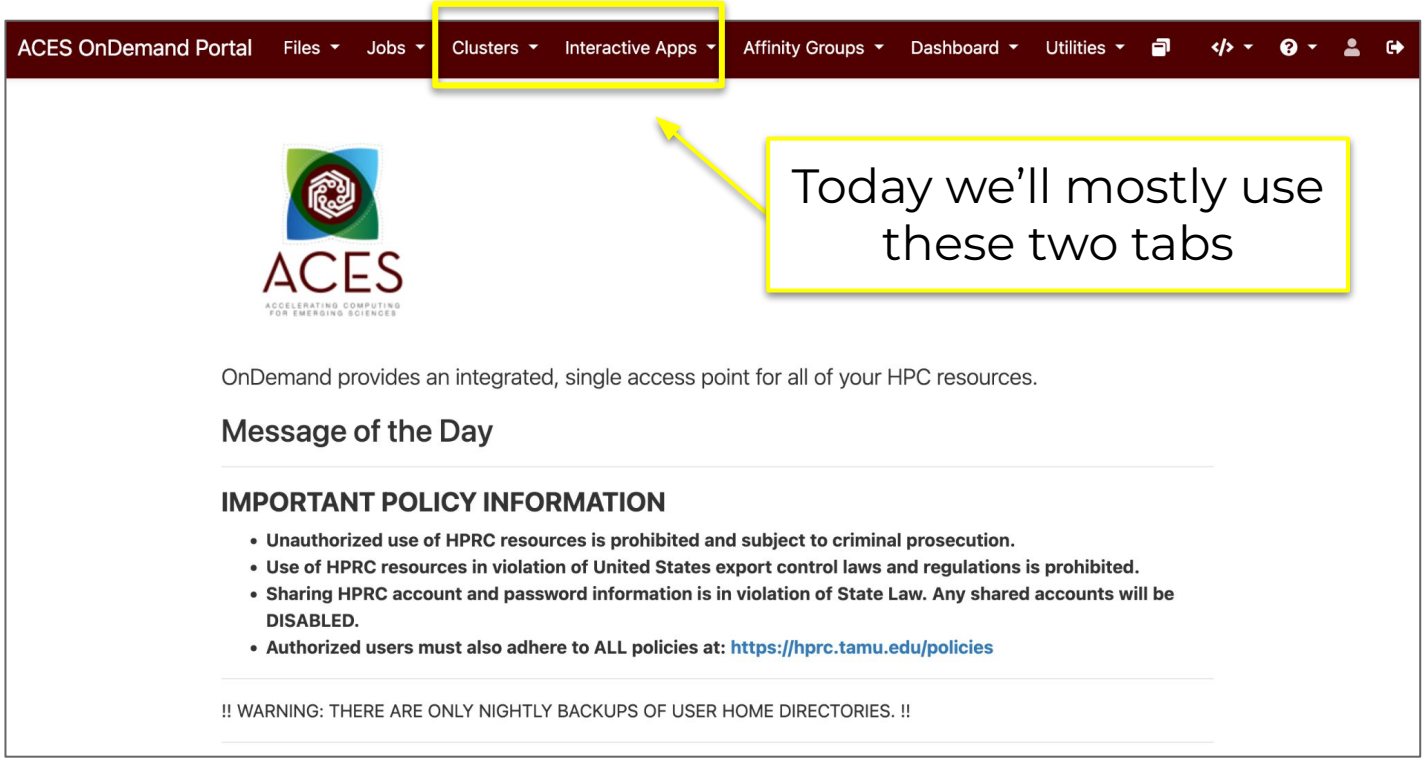


Select the Identity Provider appropriate for your account.

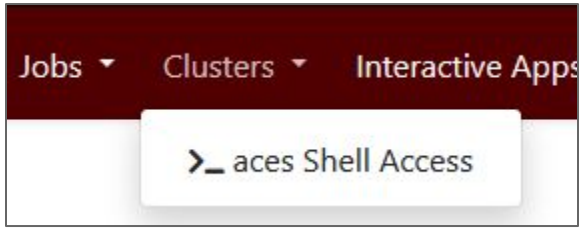


Log-in using your ACCESS or institutional credentials.

The ACES Portal



ACES Shell

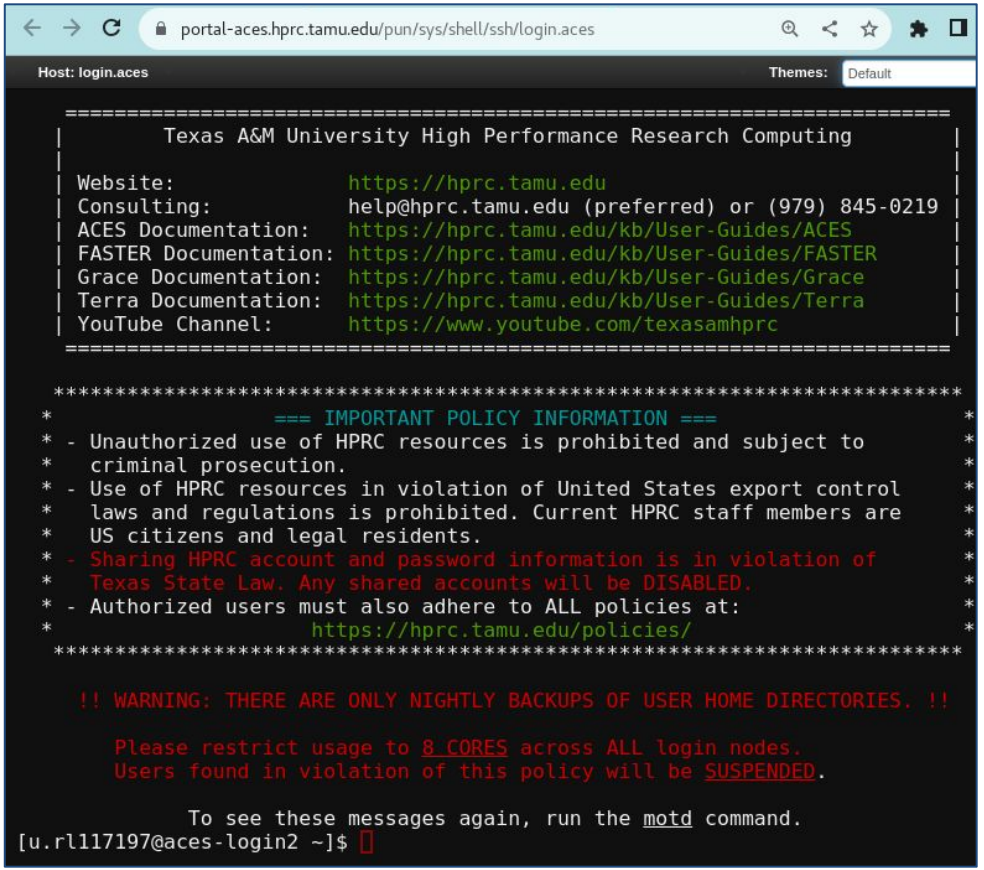


Select:

Clusters

-> _aces Shell Access

Welcome to the ACES login node.



Modules

Computing Environment

Managing software versions using Lmod

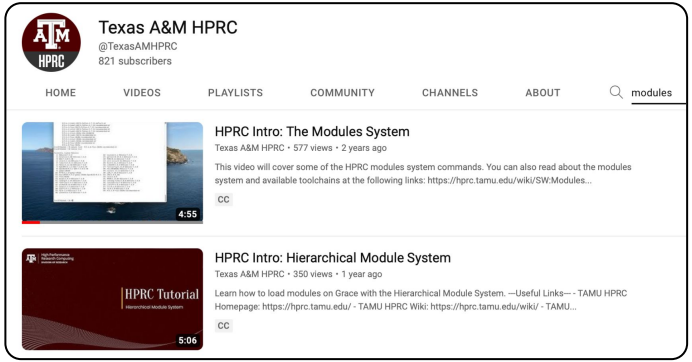
- Uses the command: **module**
- Each version of a software, application, library, etc. is available as a module.
  - Module names have the format:

toolchain-name / version toolchain-name / version

GCC/14.3.0 OpenMPI/5.0.8

software-name / version-spec

PyTorch/2.7.0-CUDA-12.6.0



- module sets the correct environment variables for you.

hprc.tamu.edu/kb/Software/useful-tools/Modules

Module Usage Basics

`module avail`

- Lists all available modules (may be slow).
- Navigation:
  -spacebar, arrows, or `j` and `k`
  -quit with `q`
- Case-sensitive search: `/`
- Use **`mla`** instead to save results to a file as well (will be named `module.avail.aces` or similar)

`module spider <word>`

- Case-sensitive search for modules with “word” in name.
- Provide an exact name to see dependencies.

```
[u.jw123527@aces-login2 ~]$ module spider Python

----------------------------------------------------------------------------------------------------------------------
  Python:
----------------------------------------------------------------------------------------------------------------------
    Description:
      Python is a programming language that lets you work more quickly and integrate your systems more effectively.

     Versions:
        Python/2.7.18-bare
        Python/2.7.18
        Python/3.8.6
        Python/3.9.5-bare
        Python/3.9.5
        Python/3.9.6-bare
        Python/3.9.6
        Python/3.10.4-bare
        Python/3.10.4
        Python/3.10.8-bare
        Python/3.10.8
        Python/3.11.3
        Python/3.11.5
     Other possible modules matches:
        Biopython  Boost.Python  Brotli-python  IPython  LASSO-Python  Python-bundle-PyPI  flatbuffers-python  graphvi

----------------------------------------------------------------------------------------------------------------------
  To find other possible module matches execute:

      $ module -r spider '.*Python.*'
```

Module Usage Basics

`module load <module>`

- add <module> paths to the current environment variables

`module list`

- see what modules are loaded in your current session

`module purge`

- unload all modules

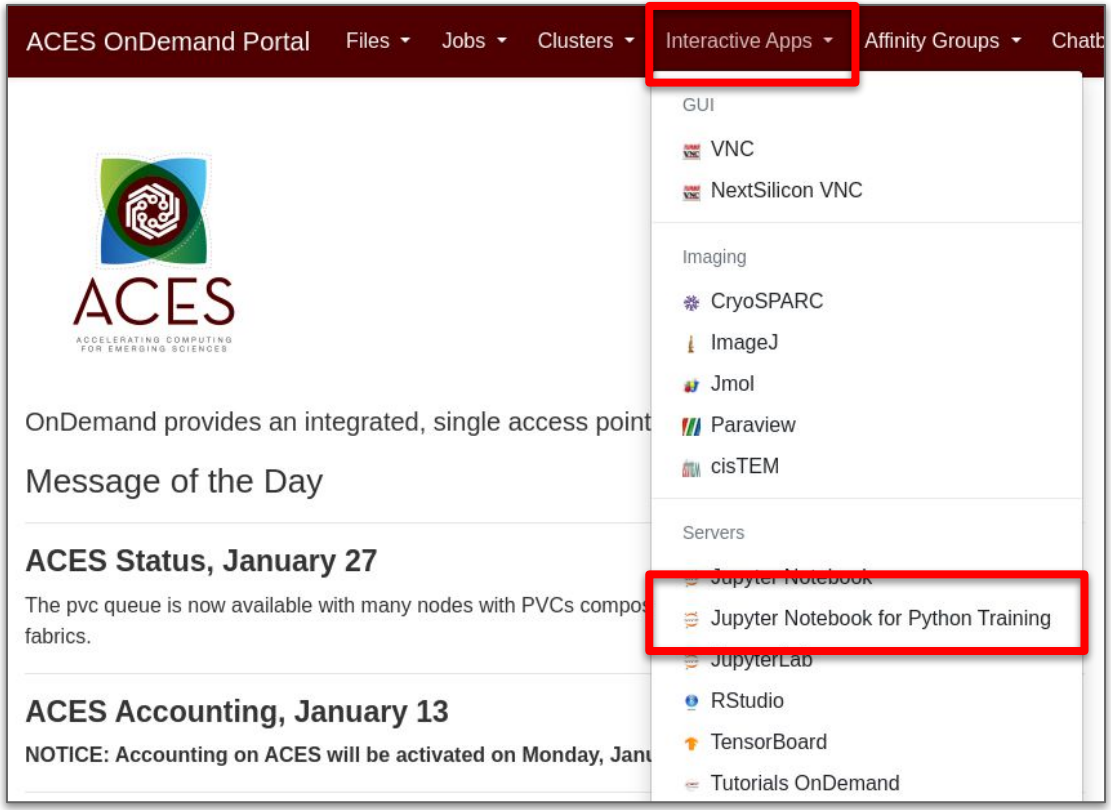
```
[u.jw123527@aces-login2 ~]$ module load GCCcore/13.2.0
[u.jw123527@aces-login2 ~]$ module list

Currently Loaded Modules:
  1) GCCcore/13.2.0



[u.jw123527@aces-login2 ~]$ module purge
[u.jw123527@aces-login2 ~]$ module list
No modules loaded
[u.jw123527@aces-login2 ~]$
```

Hands-on Exercise: Module Loading

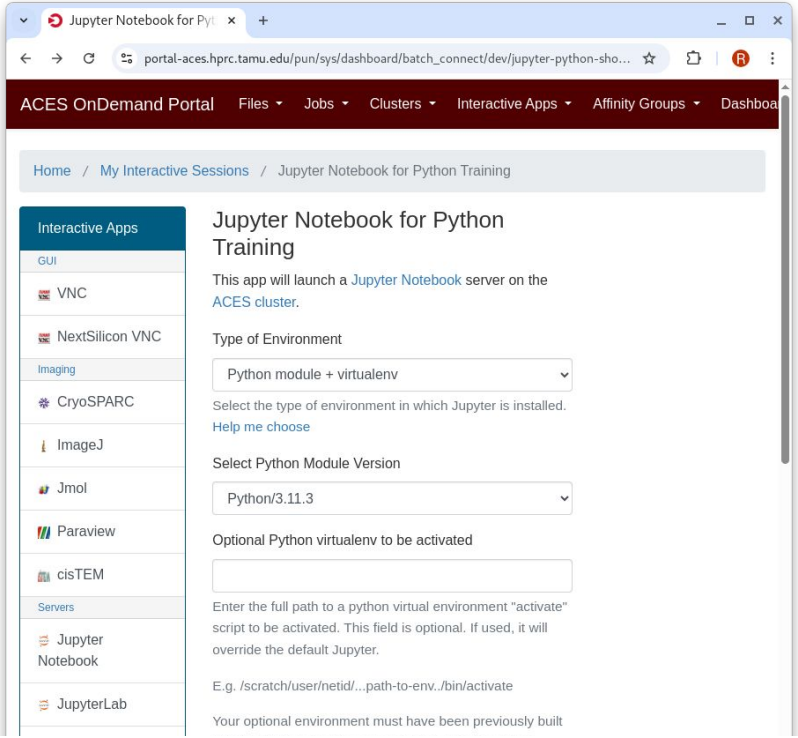
1. `mla Python` -See which versions of Python are available.
2. `ml Python/3.11.3` -error! You cannot do that yet.
3. `module spider Python/3.11.3` -Learn how to load this module.
4. `ml _____ Python/3.11.3` -Fill in the blank (with the correct toolchains) to load this module.
5. `ml list` -List all loaded modules.
6. `ml GCC/13.2.0` -Change version of a loaded Toolchain module (GCC); notice the message about reloaded modules.
7. `ml list` -List all loaded modules.
8. `ml purge` -Remove all loaded modules.

Jupyter

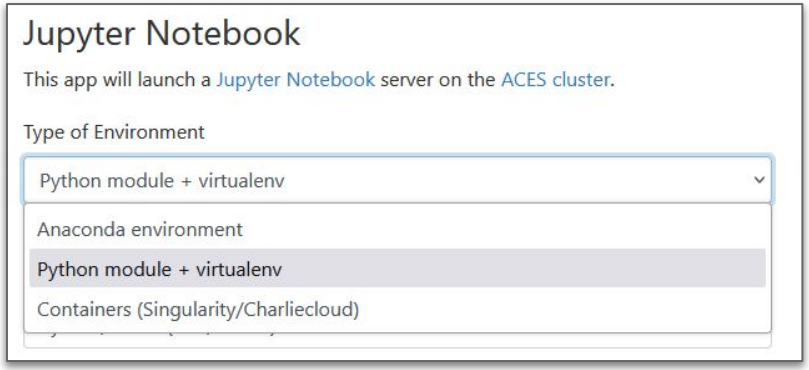
Jupyter Notebook for Python Training



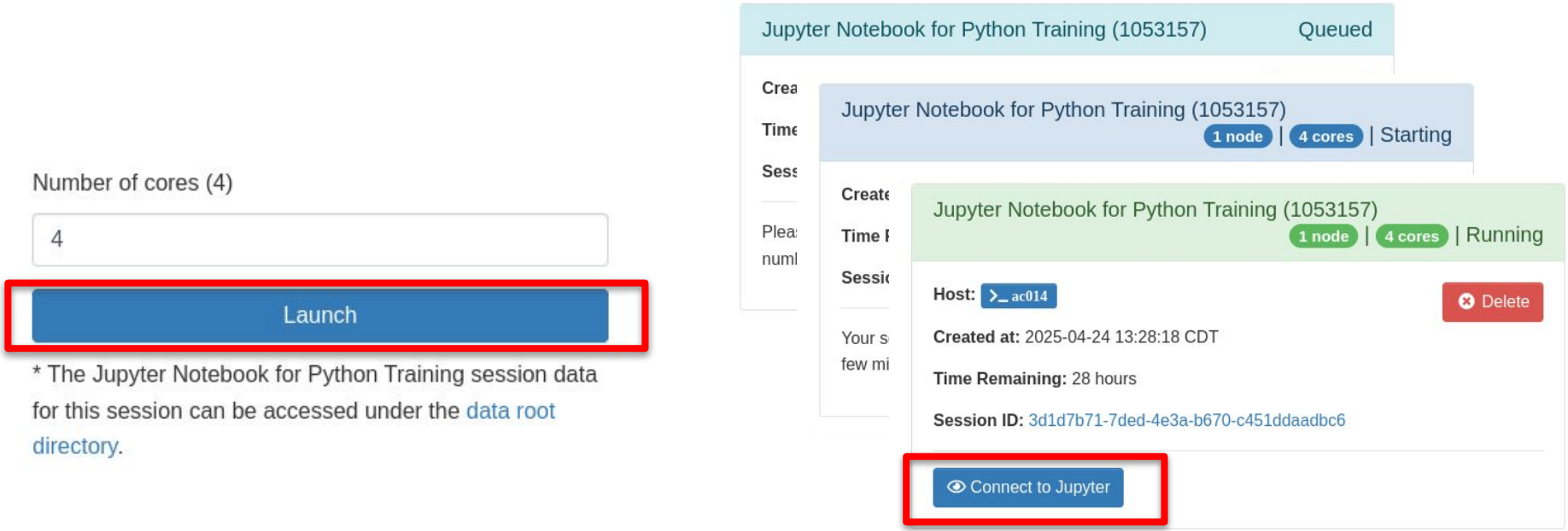
Jupyter Notebook Interactive App



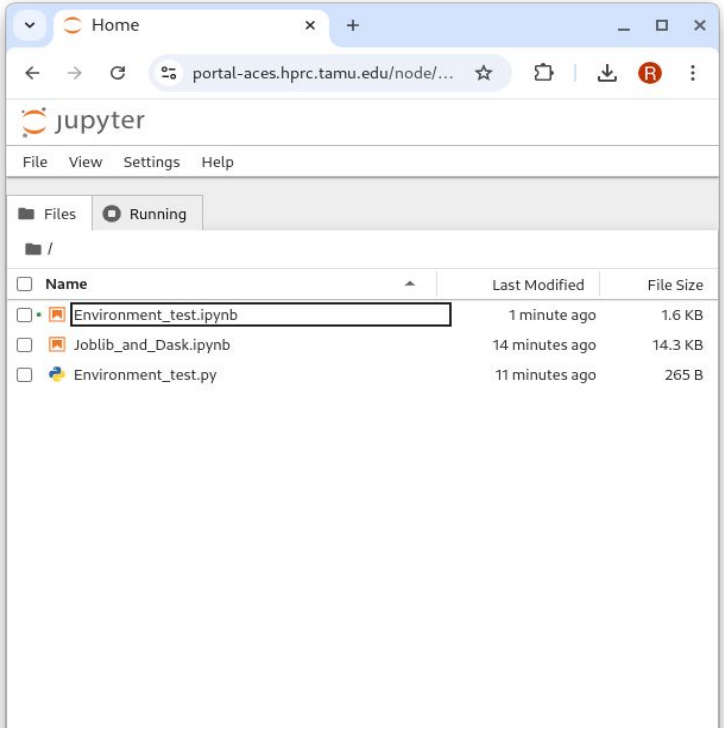
We'll come back to this form several times this class. Note the "Type of Environment" menu.

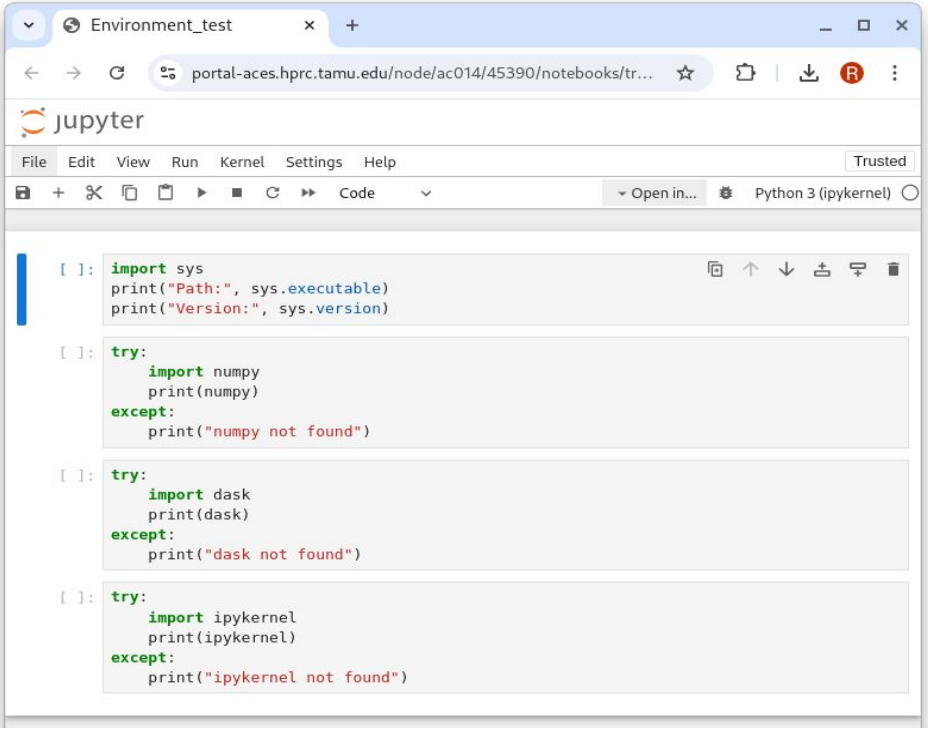


Jupyter Notebook for Python Training



Environment Test Notebook





Environment Test Script

```
cd $SCRATCH/training-2025-12-02
./Environment_test.py
Path: /usr/bin/python3
Version: 3.6.8 (default, Nov 15 2024, 08:11:39)
[GCC 8.5.0 20210514 (Red Hat 8.5.0-22)]
numpy not found
dask not found
ipykernel not found

module load foss/2022b SciPy-bundle/2023.02
./Environment_test.py
Path: /sw/eb/sw/Python/3.10.8-GCCcore-12.2.0/bin/python3
Version: 3.10.8 (main, Apr  4 2023, 12:24:51) [GCC 12.2.0]
<module 'numpy' from '/sw/eb/sw/SciPy-bundle/2023.02-gfbf-2022b/lib/python3.10/…'>
dask not found
ipykernel not found
```

Virtual Environments

Python and Virtual Environments

Python is a language which supports many external libraries in the form of extensions (called Python Packages).

Some commonly used packages:

- SciPy & NumPy
- Jupyter notebook
- Scikit-learn

You will not have these in your environment by default, but there are several ways to get them yourself.

We'll start with *virtual environments*.

See also: https://hprc.tamu.edu/kb/Software/Python/#hprc-venv-management-tools

Virtual Environments

- What is a virtual environment?
  - a file structure to hold different Python versions and package dependencies
- Why do I need one?
  - Without one, all packages are installed into your base Python Installation, which may cause conflicts across different projects.
- A good practice is to name your environment so that you can identify which Python version is in your virtualenv so that you know which module to load.
- Typically you have a virtual environment for each of your projects.

Hands-on Exercise: Python Software Install

1.
```
ml purge
ml GCCcore/12.3.0 Python/3.11.3
```
-Set up the underlying Python module

2.
```
create_venv myEnv    [-d <description>]
```
-Use HPRC's "create_venv" tool. Creates a virtual environment in $SCRATCH/virtual_envs

3.
```
source activate_venv myEnv
```
-Activate virtual environment.

4.
```
cd $SCRATCH/training-2025-04-25
./Environment_test.py
```
-Check if dask is installed (it is not).

5.
```
pip install dask
```
-Install dask.

6.
```
./Environment_test.py
```
-Where is dask installed?

7.
```
deactivate
```
-Close virtual environment.

Also Recommended

```
source activate_venv myEnv
```

```
pip install notebook numpy joblib dask ipycytoscape
```

needed for parallel exercises later

```
python3 -m ipykernel install --user --name=myEnv
```

ipykernel module from the notebook package has an additional (optional) installation step

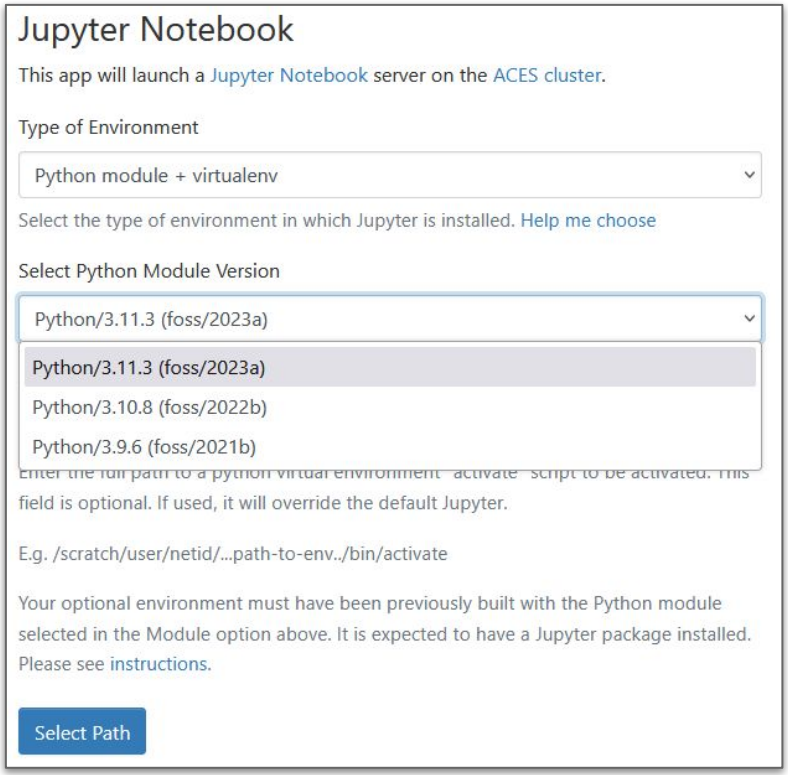
```
deactivate
```

```
list_venvs
```

Virtual Environments in Jupyter

To use your virtualenv in a Jupyter notebook:

- Select the Jupyter Notebook Interactive App in the portal.
- Choose "Python module + virtualenv".
- Select the module used to build your venv.



Virtual Environments in Jupyter

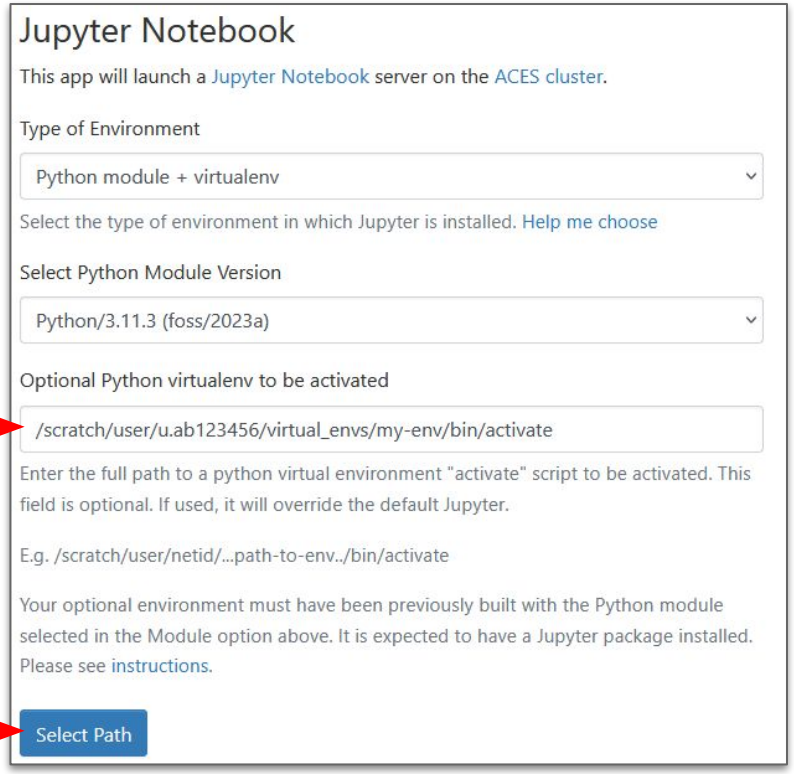
After selecting the module, provide the path to the 'activate' script of your virtual environment.

Can type it in directly...

yours (likely) at
`$SCRATCH/virtual_envs/myEnv/bin/activate`

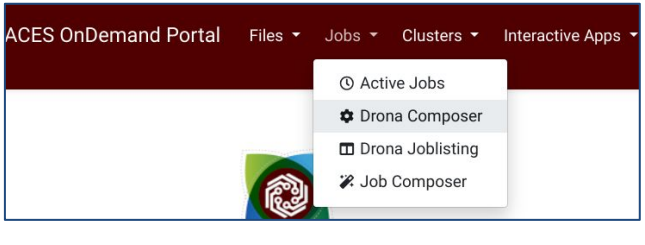
backup copy at
`$SCRATCH/training-2025-04-25/training-env/bin/activate`

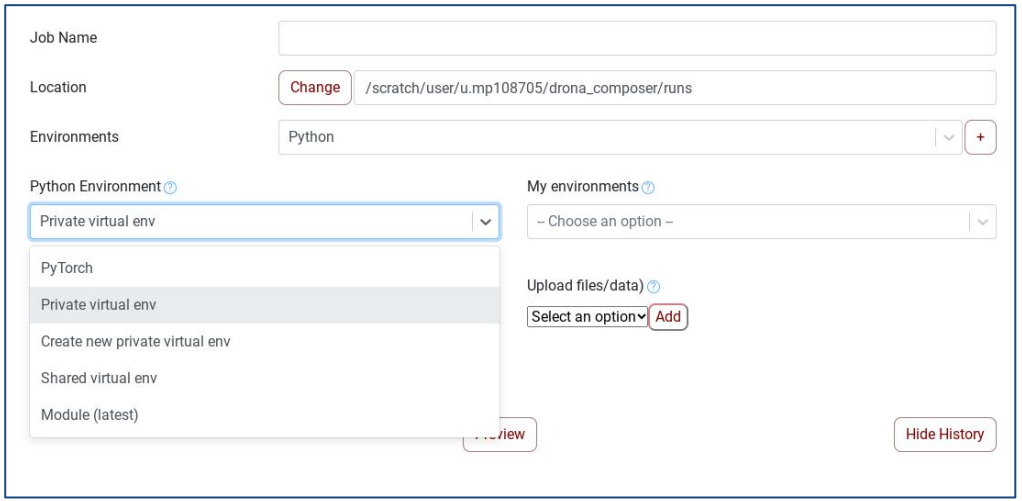
...Or select it with a mini file browser here



Running Python Codes Using Drona

- Drona assists researchers in generating and submitting (Python) jobs
- Select Python environment
- Choose your virtual env from dropdown or create one on the fly (uses create_venv)
- Select main python script and other required scripts/data.
- Drona generate and submit batch job on user's behalf





Conda Environments

[Ana]Conda

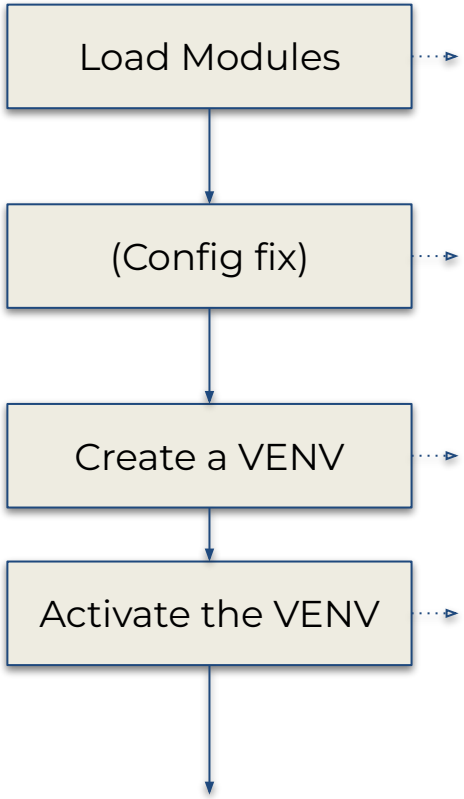
Conda is a package manager that may handle dependencies better than plain virtual environments and pip… but has more clutter.

Anaconda is a collection of packages including Conda and many other tools useful to Python programmers.

So loading an Anaconda module will give you some useful packages to start and you'll use Conda to add more.

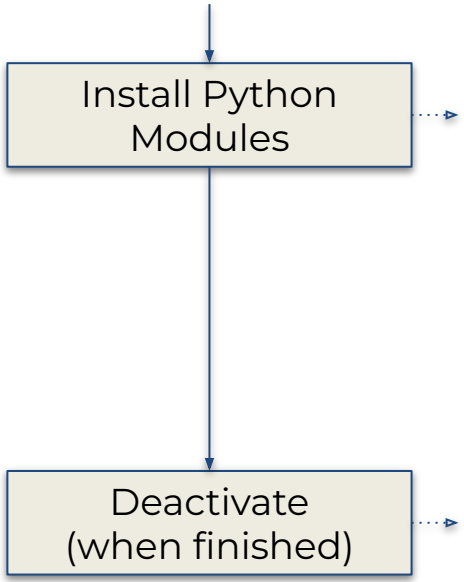
See our KB page for more information:
https://hprc.tamu.edu/kb/Software/ANACONDA/

Conda Environments




```
# clean up and load Anaconda
module purge
module load Anaconda3/2024.02-1


#1-time only: stop some Anaconda auto-loading
conda config --set auto_activate_base False


# create a Python virtual environment
conda create -n myEnv


# activate the virtual environment
source activate myEnv
```

Conda Environments

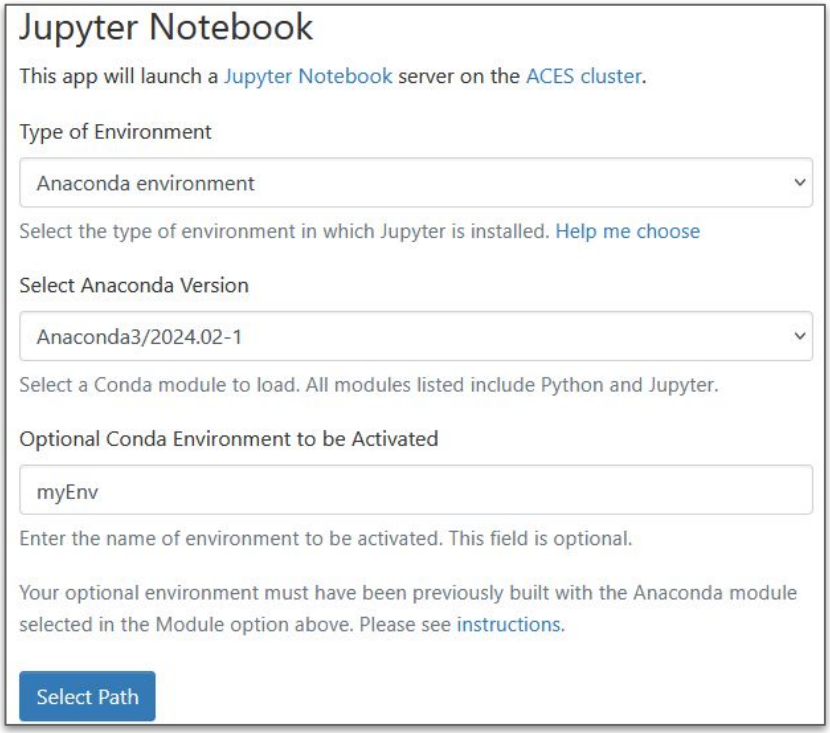



```
# install required packages for use in the portal
conda install dask
conda install -c conda-forge notebook
conda install -c conda-forge ipycytoscape


# check what we have installed
cd $SCRATCH/training-2025-12-02
./Environment_test.py


# deactivate the virtual environment
# conda deactivate
```

Anaconda Environments

To use your Anaconda environment in a Jupyter notebook:

- Select the Jupyter Notebook Interactive App in the portal.
- Choose "Anaconda environment".
- Select the module used to build your environment.
- Select your environment.



Containers

Set Up Your Singularity Environment

Get to a compute node from the login node.

```
srun --time=120 --ntasks=1 --mem=4G --pty bash -i
```

following along live? add:
`--reservation=python`

Return to your tutorial directory (if necessary).

```
cd $SCRATCH/training-2025-12-02
```

Set your singularity cache directory for temporary files.

```
export SINGULARITY_CACHEDIR=$TMPDIR
```

Connect to the internet for fetching images.

```
module load WebProxy
```

Singularity Pull

Singularity can fetch images from repositories and also convert them to the singularity file format at the same time.

```
singularity pull [target-filename] <source>
```

Where `<source>` refers to something on the internet. The syntax depends on where the source is located.

and `[target-filename]` includes the file extension.

Singularity Pull Example

The `<source>` argument for Docker images looks like

`docker://<group>/<name>[:<tag>]`

Therefore, the pull command for the Jupyter example is

```
singularity pull docker://jupyter/scipy-notebook:latest
```

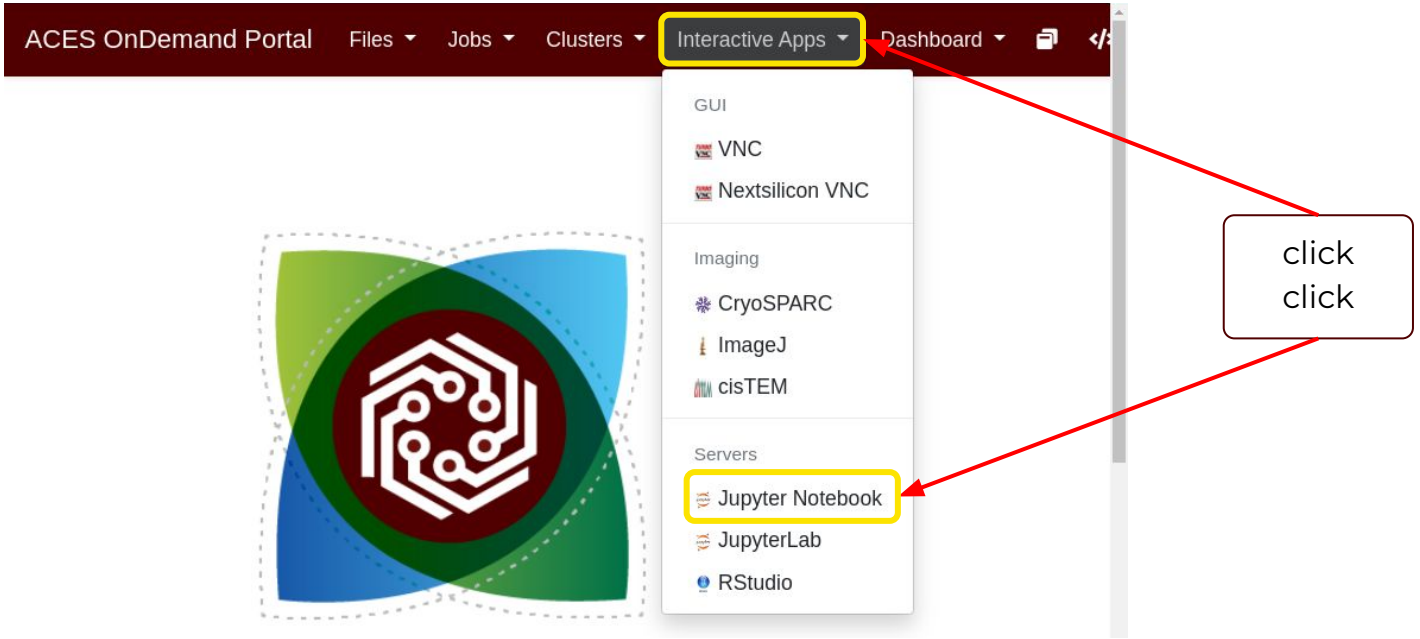
The default filename will be `scipy-notebook_latest.sif`

(Download now, or just use the backup copy linked to `/scratch/training/singularity/scipy-notebook_2023.sif`)

Singularity Environment Test

```
cd $SCRATCH/training-2025-12-02
./Environment_test.py
Path: /usr/bin/python3
Version: 3.6.8 (default, Nov 15 2024, 08:11:39)
[GCC 8.5.0 20210514 (Red Hat 8.5.0-22)]
numpy not found
dask not found
ipykernel not found

singularity exec -B /scratch scipy-notebook_2023.sif   ./Environment_test.py
singularity exec -B /scratch scipy-notebook_latest.sif ./Environment_test.py
Path: /opt/conda/bin/python3
Version: 3.11.6 | packaged by conda-forge | (main, Oct  3 2023, 10:40:35) [GCC 12.3.0]
<module 'numpy' from '/opt/conda/lib/python3.11/site-packages/numpy/__init__.py'>
<module 'dask' from '/opt/conda/lib/python3.11/site-packages/dask/__init__.py'>
<module 'ipykernel' from '/opt/conda/lib/python3.11/site-packages/ipykernel/__init__.py'>
```

Interactive Graphical Computing



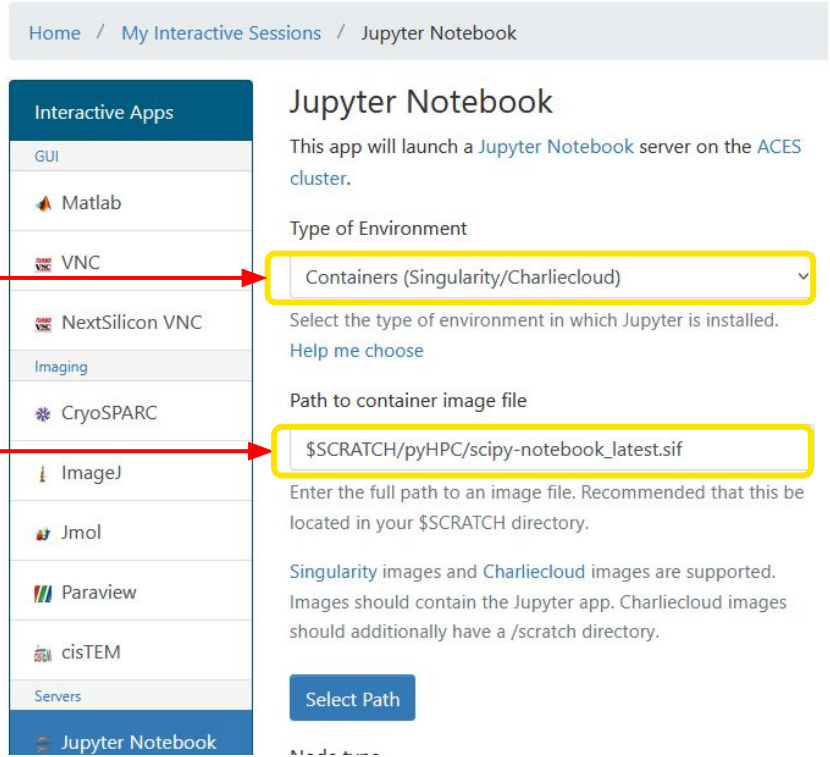
Containerized Jupyter Notebook

Choose *Containers*

Enter

`$SCRATCH/training-2025-12-02/scipy-notebook_latest.sif`

or wherever your file actually is (see Slide 34–35)



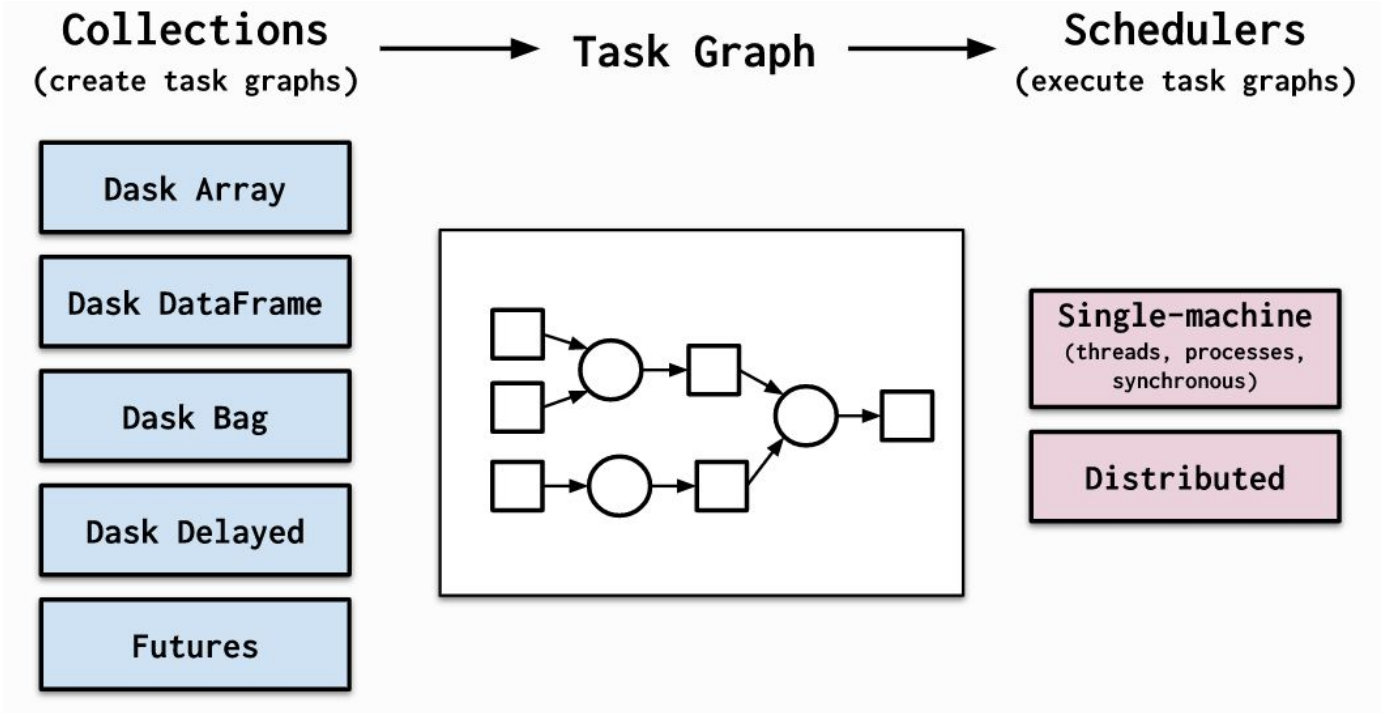
Backup copy at
`/scratch/training/singularity/scipy-notebook_2023.sif`

...Continued

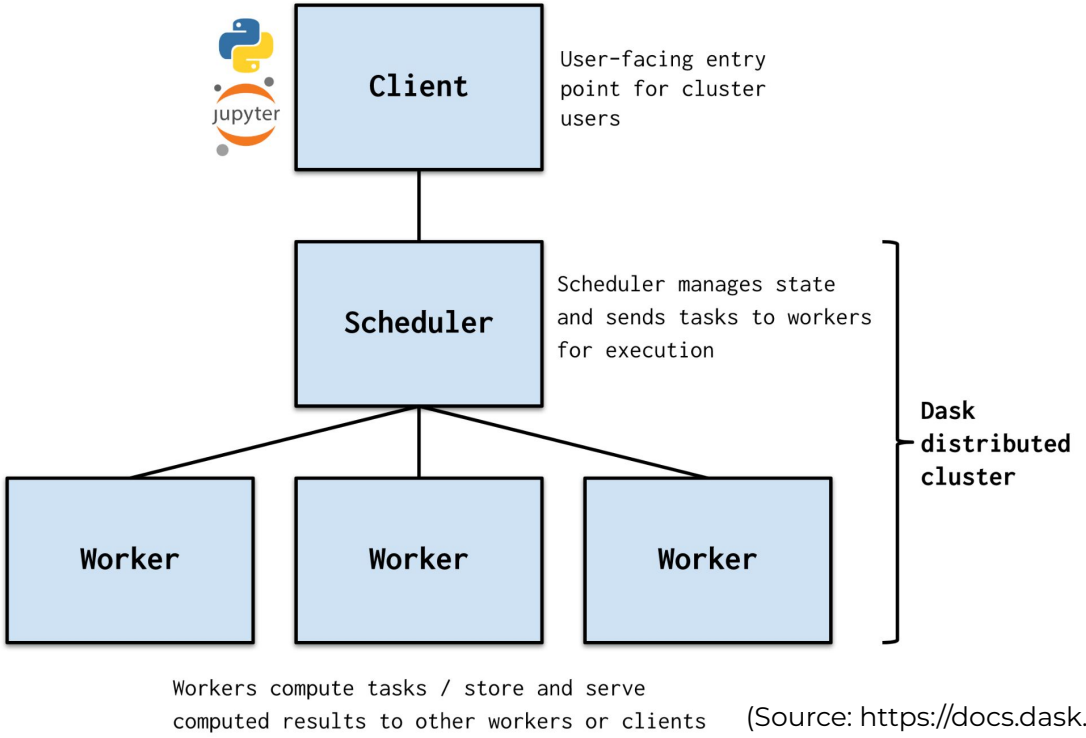

Launch
Jupyter Notebook (5488) 1 node | 1 core | Starting
Jupyter Notebook (5489) 1 node | 1 core | Running
Host: ac110
Delete
Created at: 2023-09-21 15:39:52 CDT
Time Remaining: 56 minutes
Session ID: a5f41dfd-7c0d-44e3-aea7-7331c66a4d24
Connect to Jupyter
click
...wait
click
...wait
click
New
Upload
Notebook
Terminal
Console
New File
New Folder
last month
File Edit View Run Kernel Settings Help
Trusted
Code
[1]:
import numpy
print(numpy)
WOW
<module 'numpy' from '/opt/conda/lib/python3.11/site-packages/numpy/__init__.py'>

Parallelism with Joblib and Dask

Dask Concepts



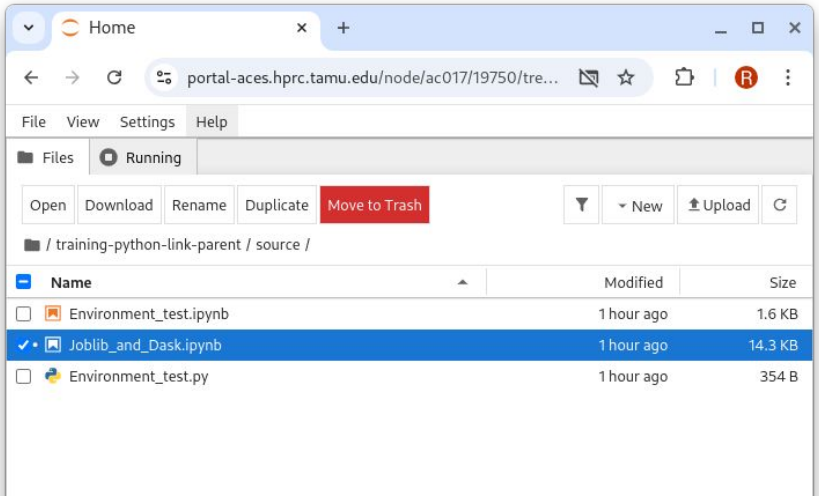
Dask Flowchart

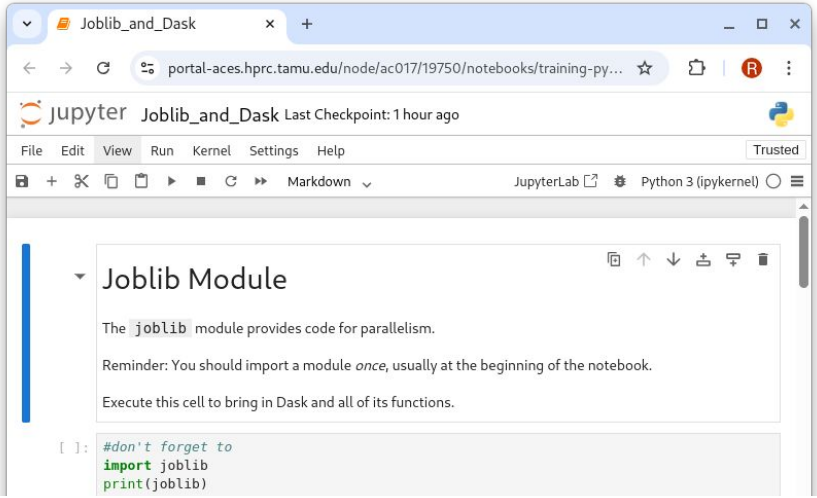


(Source: https://docs.dask.org/en/latest/)

Parallelism Exercises

- Use an environment that has the recommended modules:
  - myEnv
  - scipy-notebook_2023.sif





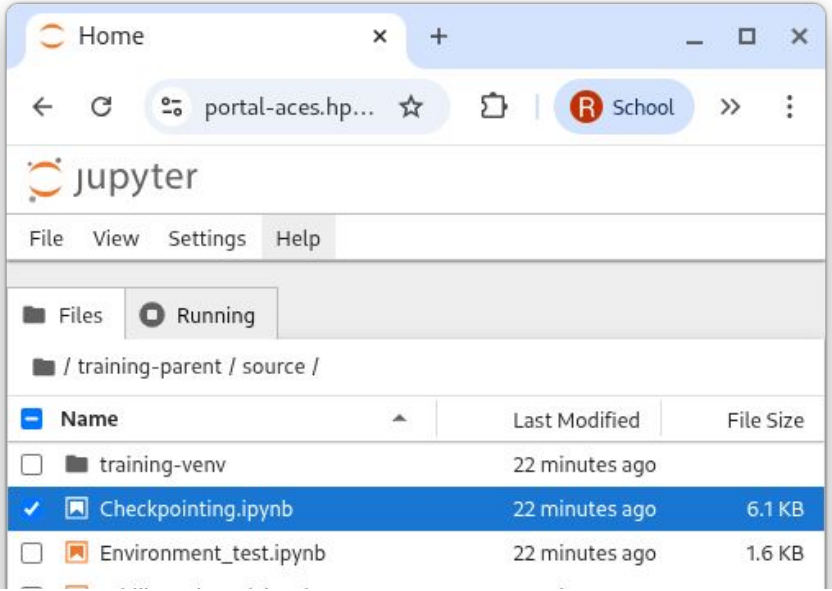
Checkpointing with Pickle

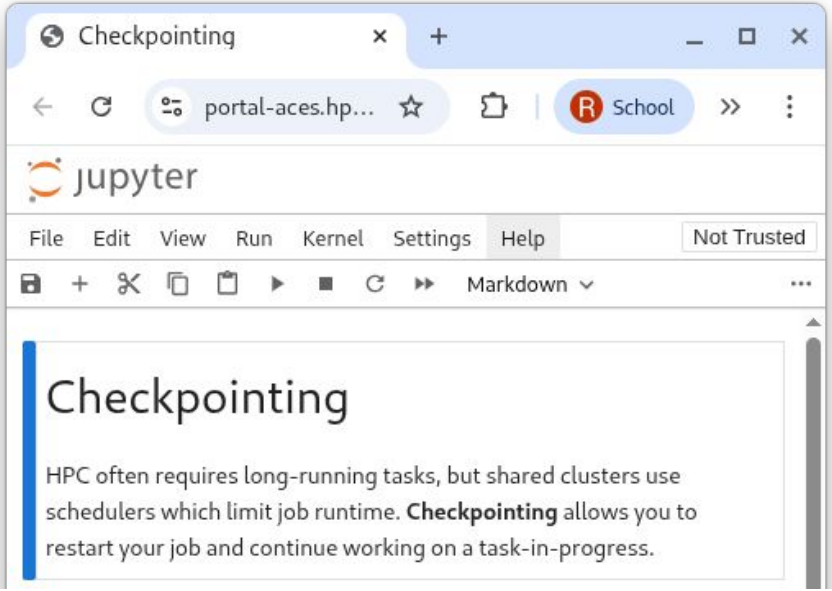
Checkpointing Concepts

- Serialization
  - Objects in memory → Binary file
- Deserialization
  - Binary file → Objects in memory

Checkpointing Exercise

- Use any environment





Acknowledgements

This work was supported by

- the National Science Foundation (NSF), award numbers:
  - 2112356 - ACES - Accelerating Computing for Emerging Sciences
  - 1925764 - SWEETER - SouthWest Expertise in Expanding, Training, Education and Research
  - 2019129 - FASTER - Fostering Accelerated Scientific Transformations, Education, and Research
- Staff and students at Texas A&M High-Performance Research Computing
- ACCESS CCEP pilot program, Tier-II

Need Help?

First check the FAQ

- Knowledge Base
- Send us a ticket using the dashboard tab on our web portal
- Email further questions to help@hprc.tamu.edu

Help us help you -- when you contact us, tell us:

- Which cluster you're using
- Your username
- Job id(s) if any
- Location of your jobfile, input/output files
- Application used, if any
- Module(s) loaded, if any
- Error messages
- Steps you have taken, so we can reproduce the problem

High Performance Research Computing

DIVISION OF RESEARCH

https://hprc.tamu.edu

HPRC Helpdesk:

help@hprc.tamu.edu
Phone: 979-845-0219

Take our short course survey!

HPRC Survey

https://u.tamu.edu/hprc_shortcourse_survey

Thank you! Any Questions?

Lunch until 1:30 pm (central)