Python and Conda Environments in HPC: *From Basics to Best Practices*

Hui (Julia) Zhao
NJIT High Performance Computing

November 5, 2025

Outline

- Why High Performance Computing
- How to access Python on Wulver at HPC
- Introduction to Conda environments
- Install, uninstall and upgrade packages
- Best Practices for managing conda environments
- Common Python libraries for scientific computing

Why High Performance Computing?

- Handling Complex Problems
- Big Data Analysis
- Speeding up Research
- Parallel Computing
- Resource Sharing and Collaboration

Why Use Python for HPC?

- **Clear Syntax** – Simple, readable, and easy to learn
- **Extensive Libraries** – Optimized packages for scientific computing
- **Multi-language Integration** – Works seamlessly with C, C++, and Fortran
- **Parallel Computing Capabilities** – Supports multi-threading & distributed computing
- **Strong Community Support** – Actively maintained & widely adopted

NJIT

Python on Wulver

Software	Version	Dependent Toolchain	Module Load Command
Python	3.13.1	foss/2025a	`module load foss/2025a Python/3.13.1`
Python	3.12.3	foss/2024a	`module load foss/2024a Python/3.12.3`

Installing Python packages

Method 1: Installing Python Packages from Source

1 Clone the repository

```
$ git clone https://github.com/pandas-dev/pandas.git
```

2 Navigate into the package directory

```
$ cd pandas
```

3 Install the package to a custom location

```
$ python setup.py install --prefix=/project/$GROUP/$USER/python_pkg/
```

Possible Installation Error

Error Message:

Traceback (most recent call last):

File "/usr/lib64/python3.6/site-packages/numpy/core/__init__.py", line 16, in <module>

from . import multiarray

ImportError: libopenblasp.so.0: cannot open shared object file: No such file or directory

Reason: The required shared library (libopenblasp.so.0) is missing or not found.

Installing Python packages - PiP

Method 2: pip

pip – stands for **“Preferred Installer Program”**
A package manager for Python packages **only**
Installs packages from the **Python Package Index (PyPI)**

$ python -m pip install --user <python-module-name> --no-cache-dir

- -m pip – Run the pip module *using this exact Python*
- --user – Install to your user account only
- --no-cache-dir – Disable pip’s cache (helpful for small quotas)

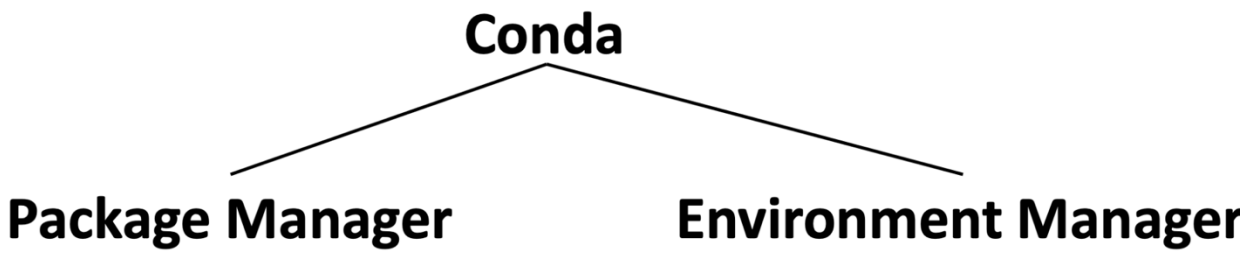
Conda on HPC

- **Introduction to Conda**
- Conda channels
- Conda environment
- Conda packages
- Sharing environments

Introduction to Conda

- **Conda is an open-source package and environment manager**
  Supports **Python and non-Python** packages
  Works across **Windows, macOS, and Linux**

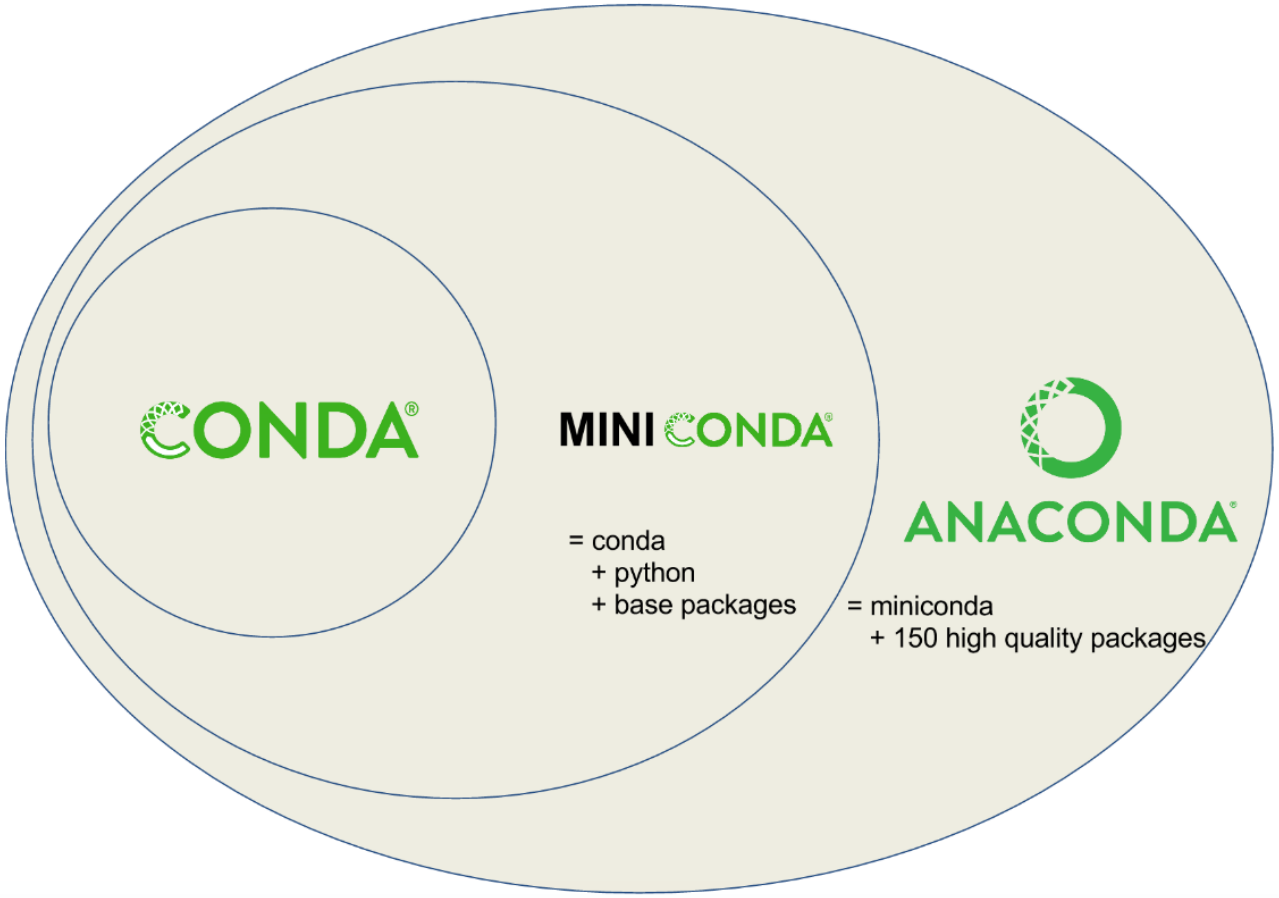
- **Conda is a powerful package & environment manager**



Why use Conda?

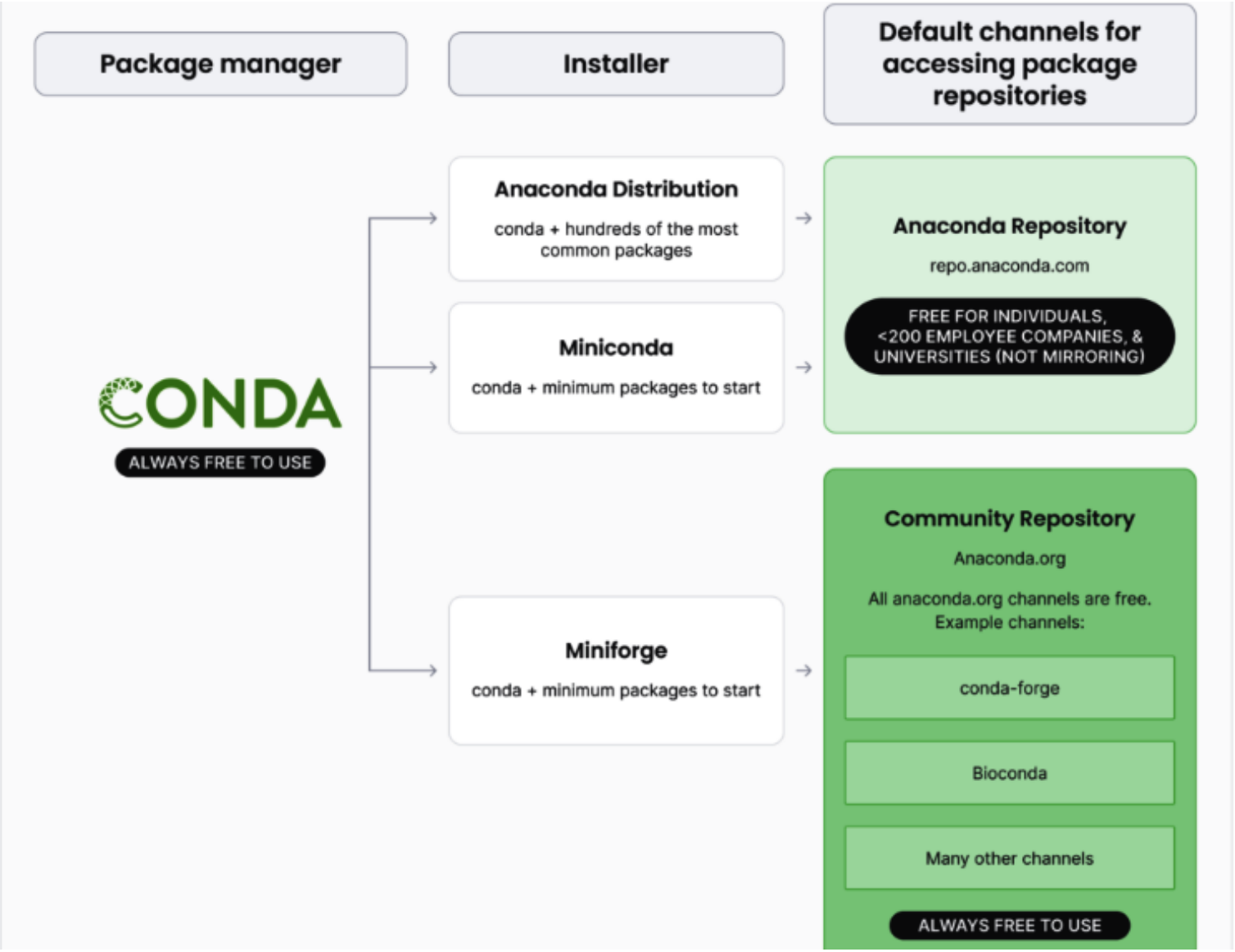
- Simplifies package management and installation
- Automatically handles dependencies
- Creates isolated environments to avoid conflicts
- Supports Python, R, C, and more
- Reproducible and portable across platforms
- Ideal for HPC and scientific computing

Anaconda vs Miniconda vs Conda



- Conda: Open-source tool
- Anaconda: A software distribution: open-source (personal) and Commercial
- Miniconda: minimal installer for conda

Anaconda Portfolio



Load Miniforge Module on Wulver

Load Miniforge3 Module

$ module load Miniforge3

Check Loaded Modules

$ module list

Currently Loaded Modules:

1) easybuild 3) slurm/wulver 5) Miniforge3/24.11.3-0

2) wulver 4) null

What is Miniforge

$ module whatis Miniforge3

Miniforge3/24.11.3-0 : Description: Miniforge is a free minimal installer for conda and Mamba specific to conda-forge.

Miniforge3/24.11.3-0 : Homepage: https://github.com/conda-forge/miniforge

Miniforge3/24.11.3-0 : URL: https://github.com/conda-forge/miniforge

Conda info

```
g07396@login02:~$ conda info

     active environment : None
            shell level : 0
       user config file : /home/g07396/.condarc
 populated config files : /apps/easybuild/el9_5.x86_64/software/Miniforge3/24.11.3-0/.condarc
                          /home/g07396/.condarc
          conda version : 24.11.3
    conda-build version : not installed
         python version : 3.12.8.final.0
                 solver : libmamba (default)
       virtual packages : __archspec=1=icelake
                          __conda=24.11.3=0
                          __glibc=2.34=0
                          __linux=5.14.0=0
                          __unix=0=0
       base environment : /apps/easybuild/el9_5.x86_64/software/Miniforge3/24.11.3-0  (read only)
      conda av data dir : /apps/easybuild/el9_5.x86_64/software/Miniforge3/24.11.3-0/etc/conda
  conda av metadata url : None
          channel URLs : https://conda.anaconda.org/conda-forge/linux-64
                          https://conda.anaconda.org/conda-forge/noarch
                          https://repo.anaconda.com/pkgs/main/linux-64
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/r/linux-64
                          https://repo.anaconda.com/pkgs/r/noarch
         package cache : /project/walsh/g07396/conda_env/pkgs_dirs
       envs directories : /home/g07396/.conda/envs
                          /apps/easybuild/el9_5.x86_64/software/Miniforge3/24.11.3-0/envs
               platform : linux-64
             user-agent : conda/24.11.3 requests/2.32.3 CPython/3.12.8 Linux/5.14.0-503.11.1.el9_5.x86_64 rhel/9.6 glibc/2.34 solver/libmamba conda-libmamba-solver/24.9.0 libmambapy/1.5.12
                UID:GID : 580857:580857
             netrc file : None
           offline mode : False
```

Conda on HPC

- Introduction to Conda
- **Conda channels**
- Conda environment
- Conda packages
- Sharing environments

Configuring Conda channels

A conda channel is a repository of conda packages

$ conda config **--help**

$ conda config **--show**

$ conda config **--show channels**

channels:
- conda-forge
- defaults

$conda config **--describe channels**

$conda config **--add channels bioconda**

This would add the conda-forge channel to the top of the channel list.

$conda config **--append** channels bioconda

This would add the bioconda to the end of the channel list, giving it the lowest priority.

Conda on HPC

- Introduction to Conda
- Conda channels
- **Conda environment**
- Conda packages
- Sharing environments

Why create a Conda environment?

A conda environment is a directory that contains a specific collection of conda packages.

Isolation from other projects

Control Over Packages

- Manage versions and dependencies.

Reproducibility

- Consistent setups across systems.

Dependency Management

- Handles Python and non-Python dependencies.

Python Versatility

- Manage and switch Python versions easily.

Cross-Platform

- Works on Linux, Windows, and macOS.

Commonly used Conda commands

Task	Command
Activate environment:	`conda activate [environment_name]`
Deactivate environment:	`conda deactivate [environment_name]`
Show the list of environments:	`conda env list`
Delete environment:	`conda remove [environment_name]`
Export environment:	`conda env export > [environment_name].yml`
Import environment from YAML:	`conda env create -f [environment_name].yml`
Import environment to different location:	`conda env create -f [environment_name].yml -p [PATH]`

Conda cheat sheet - Link to Conda Doc for more helpful commands

Creating Conda Environment

Creating a new conda environment

$ conda create --name my_env

Creating a new conda environment with a specific python version

$ conda create --name my_env python=3.9

Creating a new conda environment with a specific python version and scipy package

$ conda create --name my_env python=3.9 scipy=0.15.0

Creating a new conda environment in difference location with **--prefix** or **-p**

$ conda create --prefix /project/$GROUP/$USER/conda_env/AAA

Enter, Exit and Remove conda environment

Entering a Conda environment

$ **conda activate my_env**

$ **conda activate /project/$GROUP/$USER/conda_env/AAA**

Exiting a Conda environment we are currently in

$ **conda deactivate**

Removing a Conda environment

$ **conda env remove -n my_env**

Renaming a Conda environment

$conda rename -n old_env_name new_env_name

List conda environments

A user may list all shared virtual environments and your own private virtual environments

```
g07396@login02:~$ conda info --envs

# conda environments:
#
base                  /apps/easybuild/el9_5.x86_64/software/Miniforge3/24.11.3-0
matlab-proxy          /apps/easybuild/el9_5.x86_64/software/Miniforge3/24.11.3-0/envs/matlab-proxy
r_env                 /home/g07396/.conda/envs/r_env
tensorflow            /home/g07396/.conda/envs/tensorflow
torch-cuda            /home/g07396/.conda/envs/torch-cuda
                      /project/walsh/g07396/conda_env/torch-cuda_12.1
                      /project/walsh/g07396/conda_env/torch-cuda_12.2

g07396@login02:~$ conda env list

# conda environments:
#
base                  /apps/easybuild/el9_5.x86_64/software/Miniforge3/24.11.3-0
matlab-proxy          /apps/easybuild/el9_5.x86_64/software/Miniforge3/24.11.3-0/envs/matlab-proxy
r_env                 /home/g07396/.conda/envs/r_env
tensorflow            /home/g07396/.conda/envs/tensorflow
torch-cuda            /home/g07396/.conda/envs/torch-cuda
                      /project/walsh/g07396/conda_env/torch-cuda_12.1
                      /project/walsh/g07396/conda_env/torch-cuda_12.2
```

Conda on HPC

- Introduction to Conda
- Conda environment
- Conda channels
- **Conda packages**
- Sharing environments

Conda packages

A conda package is a compressed tarball file

```
.
├── bin
│   └── pyflakes
├── info
│   ├── LICENSE.txt
│   ├── files
│   ├── index.json
│   ├── paths.json
│   └── recipe
└── lib
    └── python3.5
```

List All Installed Packages:

- $ **conda list**
- Displays all packages installed in the active Conda environment.

List Packages in a Specific Environment:

$ **conda list -n env_name or conda list -p /path/to/environment**

Search for a Package:

- $ **conda search package_name**
- Searches for a package across all channels in Conda.

Check for Specific Package Installation:

- $ **conda list package_name –n myenv**
- $ **conda list –n myenv | grep package_name**
- Filters the list of installed packages to show only the entries related to package_name.

List packages in all environments

```
g07396@n0094:~$ conda list
# packages in environment at /apps/easybuild/el9_5.x86_64/software/Miniforge3/24.11.3-0:
#
# Name                    Version                   Build  Channel
_libgcc_mutex             0.1                 conda_forge    conda-forge
_openmp_mutex             4.5                       2_gnu    conda-forge
archspec                  0.2.5             pyhd8ed1ab_0    conda-forge
boltons                   24.0.0            pyhd8ed1ab_1    conda-forge
brotli-python             1.1.0          py312h2ec8cdc_2    conda-forge
bzip2                     1.0.8               h4bc722e_7    conda-forge
c-ares                    1.34.4              hb9d3cd8_0    conda-forge
ca-certificates           2024.12.14          hbcca054_0    conda-forge
certifi                   2024.12.14        pyhd8ed1ab_0    conda-forge
cffi                      1.17.1         py312h06ac9bb_0    conda-forge
charset-normalizer        3.4.1             pyhd8ed1ab_0    conda-forge
colorama                  0.4.6             pyhd8ed1ab_1    conda-forge
conda                     24.11.3        py312h7900ff3_0    conda-forge
conda-libmamba-solver     24.9.0            pyhd8ed1ab_0    conda-forge
conda-package-handling    2.4.0             pyh7900ff3_2    conda-forge
conda-package-streaming   0.11.0            pyhd8ed1ab_0    conda-forge
distro                    1.9.0             pyhd8ed1ab_1    conda-forge
fmt                       11.0.2              h434a139_0    conda-forge
```

List packages in an environment

```
g07396@n0094:~$ conda list -n tensorflow
# packages in environment at /home/g07396/.conda/envs/tensorflow:
#
# Name                    Version                   Build  Channel
_libgcc_mutex             0.1                 conda_forge    conda-forge
_openmp_mutex             4.5                       2_gnu    conda-forge
bzip2                     1.0.8               h4bc722e_7    conda-forge
ca-certificates           2025.1.31           hbd8a1cb_1    conda-forge
certifi                   2025.1.31         pyhd8ed1ab_0    conda-forge
ld_impl_linux-64          2.43                h712a8e2_4    conda-forge
libffi                    3.4.6               h2dba641_0    conda-forge
libgcc                    14.2.0              h767d61c_2    conda-forge
libgcc-ng                 14.2.0              h69a702a_2    conda-forge
libgomp                   14.2.0              h767d61c_2    conda-forge
liblzma                   5.6.4               hb9d3cd8_0    conda-forge
libnsl                    2.0.1               hd590300_0    conda-forge
```

List the installed packages for the present environment

(myenv) $ conda list

Installing Conda packages

- Entering a Conda environment
  - $ **conda activate my_env**
  - (my_env) $: **conda install scipy=1.6 --channel conda-forge**

- Create an environment called my_biowork-env and install blast from the bioconda channel:
  - $ **conda create --name my_biowork-env blast --channel bioconda**

- The name flag can be used to specify the environment in which we install the package
  - $ **conda install -n my_env scipy**

$**conda install conda-forge::tensorflow --prefix /project/$GROUP/$USER/my_env**

Example - Install PyTorch with GPU

```
$conda create --name torch-cuda python=3.10
$conda activate torch-cuda
$conda install -c "nvidia/label/cuda-12.2.0" cuda-toolkit
$conda install -c pytorch -c nvidia pytorch torchvision torchaudio pytorch-cuda -y
```

A simple PyTorch test program is given below to check whether PyTorch has been installed properly. Program is called

torch_tensor.py

User can use the following job script to run the script.

torch-cuda.submit.sh

https://hpc.njit.edu/Software/programming/python/conda/#install-tensorflow-with-gpu

Mamba

Mamba is a reimplementation of the conda package manager in C++ for maximum efficiency

- Parallel downloading of repository data and packages files using multi-threading
- Libsolv for much faster dependency solving
- a *drop-in* replacement for conda
- Same commands as conda
- Robust and fast but not 100% drop-in replacement yet (especially for conda-env commands)

https://mamba.readthedocs.io/en/latest/

Mamba on Wulver

```
module load Miniforge3

# create new environment
mamba create --name env_name python numpy pandas
# install a new package into an existing environment
conda activate env_name
mamba install scipy
```

Mamba on wulver

Conda on HPC

- Introduction to Conda
- Conda environment
- Conda channels
- Conda packages
- **Sharing environments**

Exporting Conda environment

Export a conda environment to a new directory or a different machine

1. activate the environment first that you intend to export.
2. export it to a YAML file:

$ conda env export > my_environment.yml

```
name: my_env
channels:
- defaults
dependencies:
- _libgcc_mutex=0.1=main
- _openmp_mutex=5.1=1_gnu
- blas=1.0=mkl

<ouput snipped>

#the last line is the path of the env
prefix: /home/a/abc3/.conda/envs/my_env.
```

Create an Conda environment from YAML file

- First load Miniforge
- Create the environment from the YAML file:

```
conda env create -f my_environment.yml
Collecting package metadata (repodata.json): done
Solving environment: done

<ouput snipped>

Downloading and Extracting Packages
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
# $ conda activate my_env
#
# To deactivate an active environment, use
#
# $ conda deactivate
```

Importing Conda environment to a new location

Use the **--prefix** or **-p** option to **specify the environment location**

- $ **conda env create -f my_environment.yml -p /project/$GROUP/$USER/conda_env/my_env**
- This will create the environment in the specified directory instead of the default conda environment directory.

Provide the full path of the environment to activate it.

- $ **conda activate /project/$GROUP/$USER/conda_env/my_env**
- $ **conda env list**

```
# conda environments:
base        /mmfs1/apps/easybuild/software/Miniforge3/24.11.3-0
            * /project/$GROUP/$USER/conda_env/my_env
```

Updating a Conda environment

When to Update:

A core dependency has a new version
You need to add a new package (e.g., for analysis or visualization)
You want to remove outdated packages or switch to better ones

Example: Update a Single Package

$ **conda update -n myenv scipy**

Update from environment.yml

$ **conda env update --name=foo --file=environment.yml --prune**
--prune removes packages no longer required and keeps your environment clean

Tip:

Always save and version-control your environment.yml file to track changes.

Conda Revision and history

Task	Command	Description
List revisions	conda list --revisions	Show numbered history of changes
Roll back	conda install --revision <n>	Restore environment to revision <n>
Export current state	conda env export > env.yaml	Backup environment definition

Best practices

Use interactive sessions on a compute node

- Use an interactive session on a compute node
- **$ srun -p general -n 1 --qos=standard --account=$PI_ucid --mem-per-cpu=2G --time=59:00 --pty bash**
- **$ interactive -a $PI_UCID -q standard -j cpu**

Use /project directory with large quotas

- Use /project/$PI/$USER directory other than the home directory for conda environments and packages.
- Managing Conda Cache and changing the default caching behavior

Avoid installing packages into your base Conda environment

Conda Cache

Default Location: **$HOME/.conda/pkgs**

Check Current Cache Directory: **conda info**

Change Cache Location**:**

- Edit **.condarc**
  pkgs_dirs:
  - /path/to/desired/cache/directory
- Use Conda Command:
  conda config --add pkgs_dirs /project/$GROUP/$USER/conda_env/pkgs_dirs
- Set Environment Variable:
  export CONDA_PKGS_DIRS=/path/to/desired/cache/directory

Clean Cache and Unused Packages

- **conda clean --tarballs**
- **conda clean --all** (remove index cache, lock files, unused cache pkgs, tarballs, logs)

More Options: **official .condarc user guide**

PiP vs Conda

✅ **Favor Conda over Pip** whenever possible
✅ **Use Conda first**, then Pip only if necessary

Why Choose Conda?

- **Pre-compiled packages** – No need to build from source
  Automatic dependency resolution – Handles package conflicts
  Better for scientific computing – Optimized for numerical libraries

When to Use Pip?

- If the package **is not available** in Conda
  If you need **the latest version** of a package $ pip install latest-package

Pip drawbacks

- Dependencies may need manual resolution
  Possible compatibility issues with Conda-installed packages

Pip installs in a Conda environment

Recommend

✅ Use Conda environments for isolation
✅ Always install Conda packages first, then use pip
⚠ Avoid installing Conda packages after using pip

Create and activate a Conda environment

$ **conda create --name my_env pandas**
$ **conda activate my_env**
Install additional packages with **pip**
(my_env)$ **python -m pip install multiregex**

Recreate the environment if you need to modify packages after using pip

- once pip has been used conda will be unaware of the changes
- to install additional conda packages it is best to recreate the environment

Store conda and pip requirements in text files

⚠ **Use --no-cache-dir** to prevent pip from filling your home directory
(my_env)$ **python -m pip install --no-cache-dir package_name**

Refer to [Conda guide for using pip in a Conda environment]

Common Python libraries for scientific computing

Library	Key features	Common Use Case
Numpy	Multidimensional arrays, Broadcasting, Vectorization	Mathematical operations, Basic statistics
SciPy	Numerical integration, Optimization, Linear algebra	Solving differential equations, Signal processing
Matplotlib	2D and 3D plotting, Customizable plots	Visualizing data, Scientific charts
Pandas	DataFrame and Series, Data manipulation, Cleaning	Data analysis, Time series analysis
Scikit-learn	Machine learning algorithms, Data preprocessing tools	Classification, Regression, Clustering
TensorFlow	Computational graph, Automatic differentiation	Building deep learning models, Neural networks
PyTorch	Dynamic computational graph, TorchScript for deployment	Machine learning, Computer vision

Connect with Us

- Open a ticket using email: hpc@njit.edu
- Request Software: HPC Software Installation
- Consult with Research Computing Facilitator: HPC User Assistance
- Further information: HPC at NJIT
- Conda on Wulver
- System updates
    - Read Message of the Day on login
    - Visit NJIT HPC News

Questions?