Introduction to Python and Conda on HPC

Hui (Julia) Zhao

NJIT High Performance Computing

New Jersey Institute of Technology

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

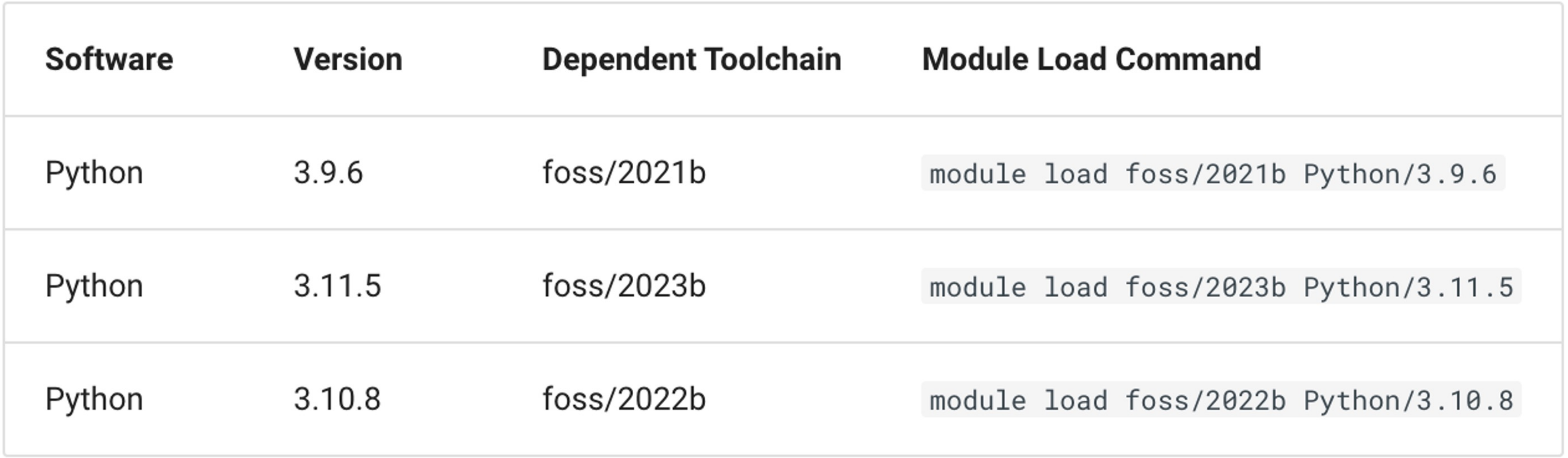
Why High Performance Computing?

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

Python in High Performance Computing

- Clear Syntax
- Extensive Libraries
- Multi-language Integration
- Parallel Computing Capabilities
- Strong Community Support

Python on Wulver



Software	Version	Dependent Toolchain	Module Load Command
Python	3.9.6	foss/2021b	`module load foss/2021b Python/3.9.6`
Python	3.11.5	foss/2023b	`module load foss/2023b Python/3.11.5`
Python	3.10.8	foss/2022b	`module load foss/2022b Python/3.10.8`

Installing Python packages

Method 1: Installing Python Packages from Source

```
python setup.py install --prefix=</path/to/install/location>
```

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

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

```
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
```

Installing Python packages Cont.

Method 2: pip

- pip stands for "preferred Installer Program"
- a package manager for Python packages only
- pip installs packages that are hosted on the Python Package Index or PyPI
- **python -m pip install --user <python module name> --no-cache-dir**
  -m *<module-name>**,** always use "python -m pip". It executes pip using the Python interpreter you specified as python*

  --user *flag tells pip to install to the user's $HOME directory, where users have full permissions.*

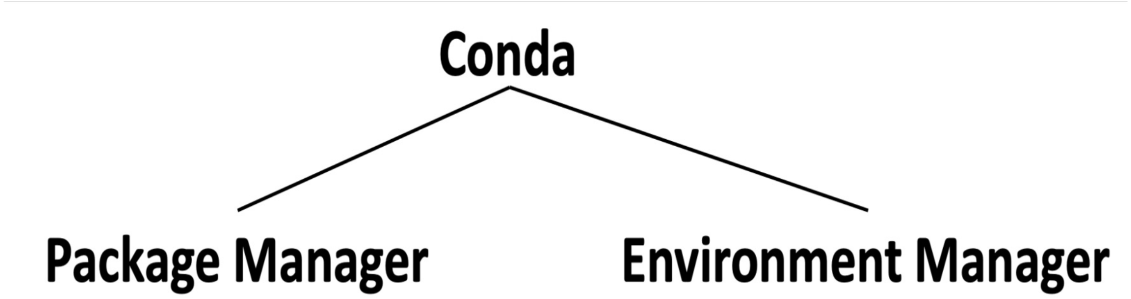
Method 3: Conda

Conda on HPC

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

Introduction to Conda

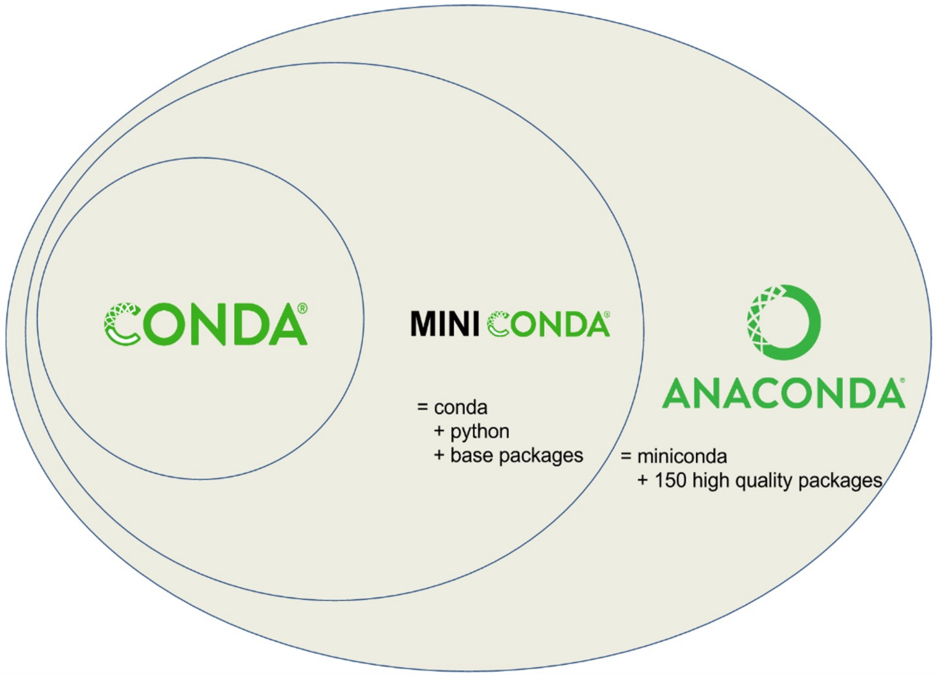
- What is Conda?
  - A package, dependency, and environment management system.
  - Suitable for multiple languages, predominantly Python.



Why use Conda?

- Why use Conda?
  - Simplifies package management and deployment.
  - Ensures consistent environments.

Anaconda vs Miniconda vs Conda



Anaconda vs Miniconda

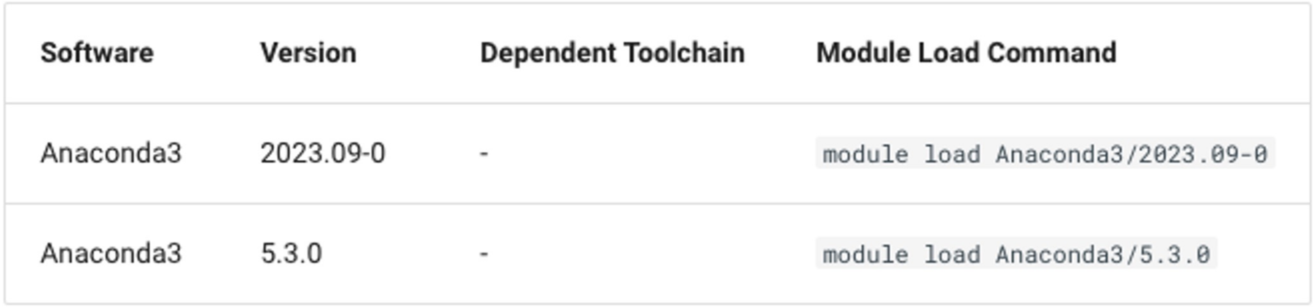
Anaconda and Miniconda are both Python distributions that come with a package manager called Conda.

Anaconda is a more comprehensive distribution than Miniconda. It comes with over 150 pre-installed packages, including many popular data science libraries such as NumPy, SciPy, and Pandas. This makes it a good choice for beginners who want to get started with data science quickly.

Miniconda is a smaller, more lightweight distribution than Anaconda. It only comes with Conda and a few other essential packages. This makes it a good choice for experienced users who want to have more control

Conda is a powerful tool that allows you to install, update, and remove Python packages.

Load the Anaconda Module on Wulver



Software	Version	Dependent Toolchain	Module Load Command
Anaconda3	2023.09-0	-	`module load Anaconda3/2023.09-0`
Anaconda3	5.3.0	-	`module load Anaconda3/5.3.0`

Use ‘module list’ to check if the correct modules are loaded

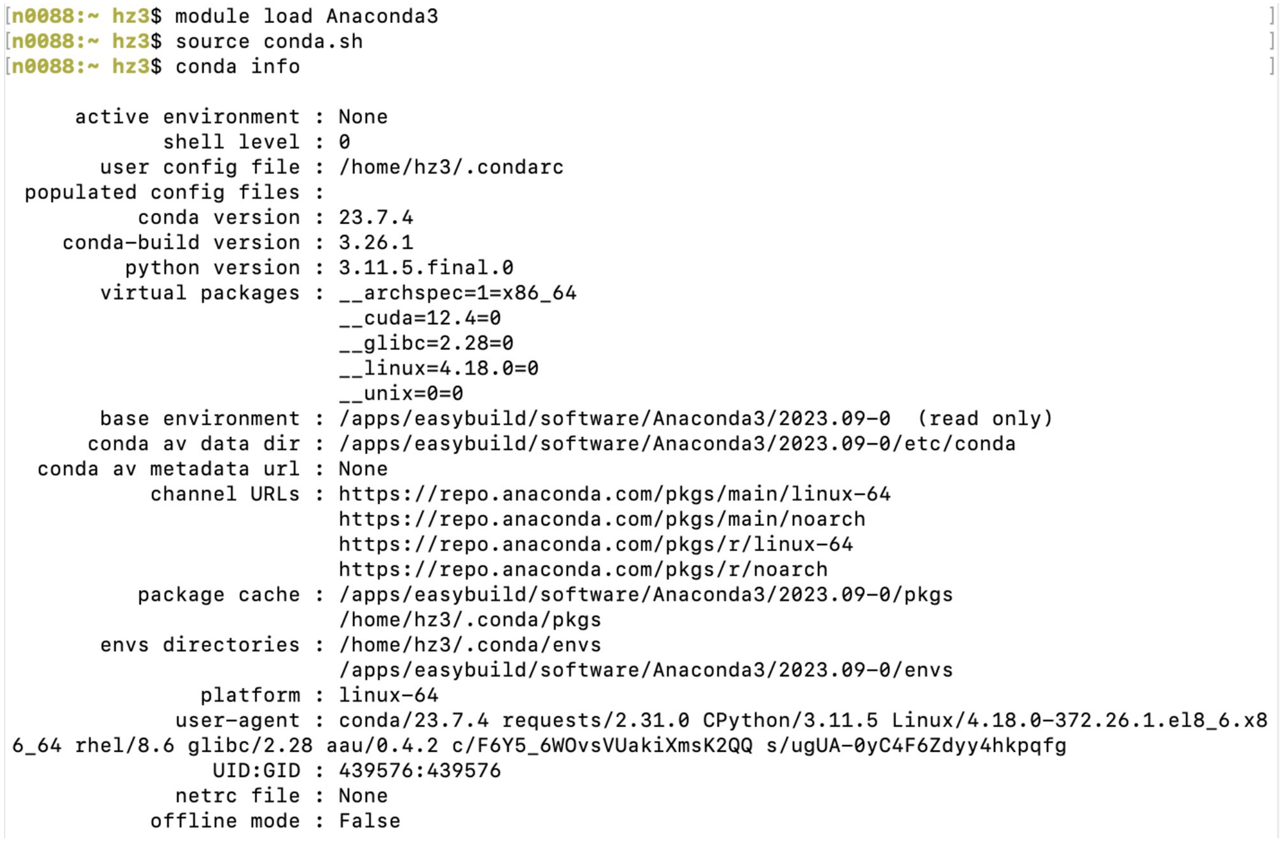
What is Anaconda

```
$ module whatis Anaconda3
```

```
Anaconda3/2023.09-0 : Description: Built to complement the rich, open source Python community,
the Anaconda platform provides an enterprise-ready data analytics platform
that empowers companies to adopt a modern open data science analytics architecture.

Anaconda3/2023.09-0 : Homepage: https://www.anaconda.com
Anaconda3/2023.09-0 : URL: https://www.anaconda.com
```

Conda info



```
[n0088:~ hz3$ module load Anaconda3                                                          ]
[n0088:~ hz3$ source conda.sh                                                                ]
[n0088:~ hz3$ conda info                                                                     ]

     active environment : None
            shell level : 0
       user config file : /home/hz3/.condarc
 populated config files :
          conda version : 23.7.4
    conda-build version : 3.26.1
         python version : 3.11.5.final.0
       virtual packages : __archspec=1=x86_64
                          __cuda=12.4=0
                          __glibc=2.28=0
                          __linux=4.18.0=0
                          __unix=0=0
       base environment : /apps/easybuild/software/Anaconda3/2023.09-0  (read only)
      conda av data dir : /apps/easybuild/software/Anaconda3/2023.09-0/etc/conda
  conda av metadata url : None
           channel URLs : 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 : /apps/easybuild/software/Anaconda3/2023.09-0/pkgs
                          /home/hz3/.conda/pkgs
       envs directories : /home/hz3/.conda/envs
                          /apps/easybuild/software/Anaconda3/2023.09-0/envs
               platform : linux-64
             user-agent : conda/23.7.4 requests/2.31.0 CPython/3.11.5 Linux/4.18.0-372.26.1.el8_6.x8
6_64 rhel/8.6 glibc/2.28 aau/0.4.2 c/F6Y5_6WOvsVUakiXmsK2QQ s/ugUA-0yC4F6Zdyy4hkpqfg
                UID:GID : 439576:439576
             netrc file : None
           offline mode : False
```

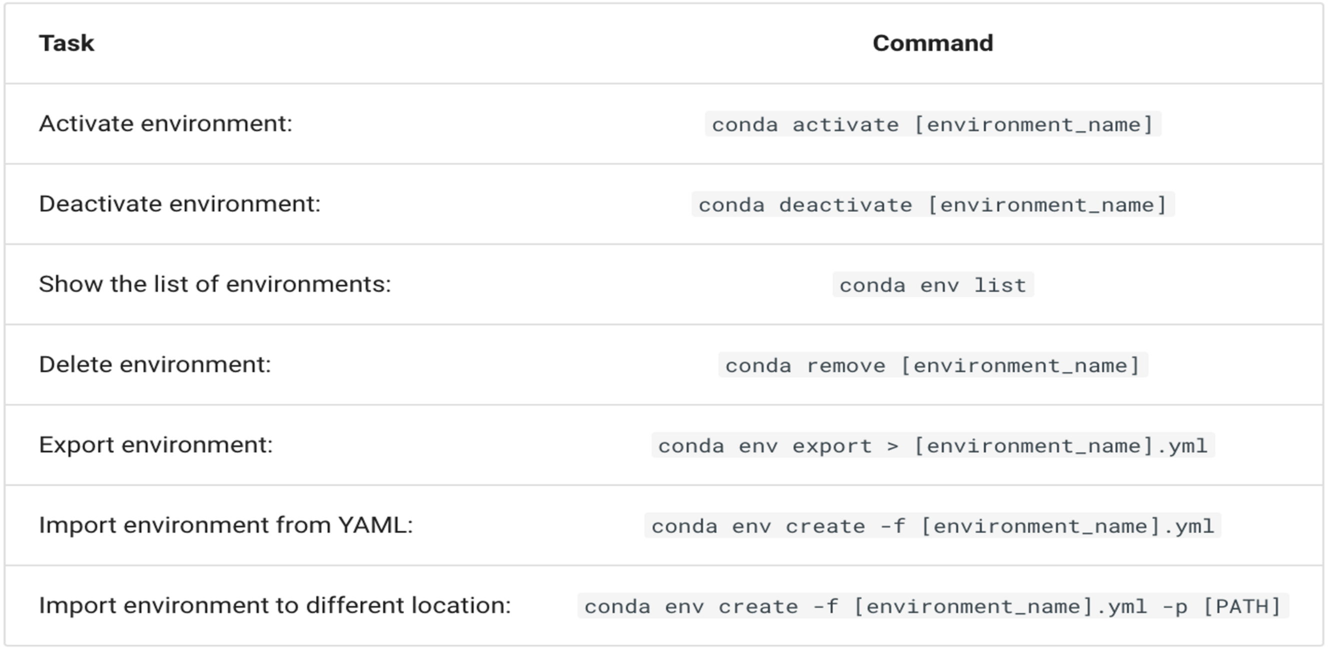
Conda on HPC

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

Why create a Conda environment?

1. **Isolation** from other projects
2. **Control Over Packages**
   - Manage versions and dependencies.
3. **Reproducibility**
   - Consistent setups across systems.
4. **Dependency Management**
   - Handles Python and non-Python dependencies.
5. **Python Versatility**
   - Manage and switch Python versions easily.
6. **Ease of Use**
   - User-friendly commands for project management.
7. **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/env_ABC AAA
```

Enter, Exit and Remove conda environment

Entering a Conda environment

```
$ conda activate my_env
(my_env) $:
$ conda activate /project/$GROUP/$USER/env_ABC
```

Exiting a Conda environment we are currently in

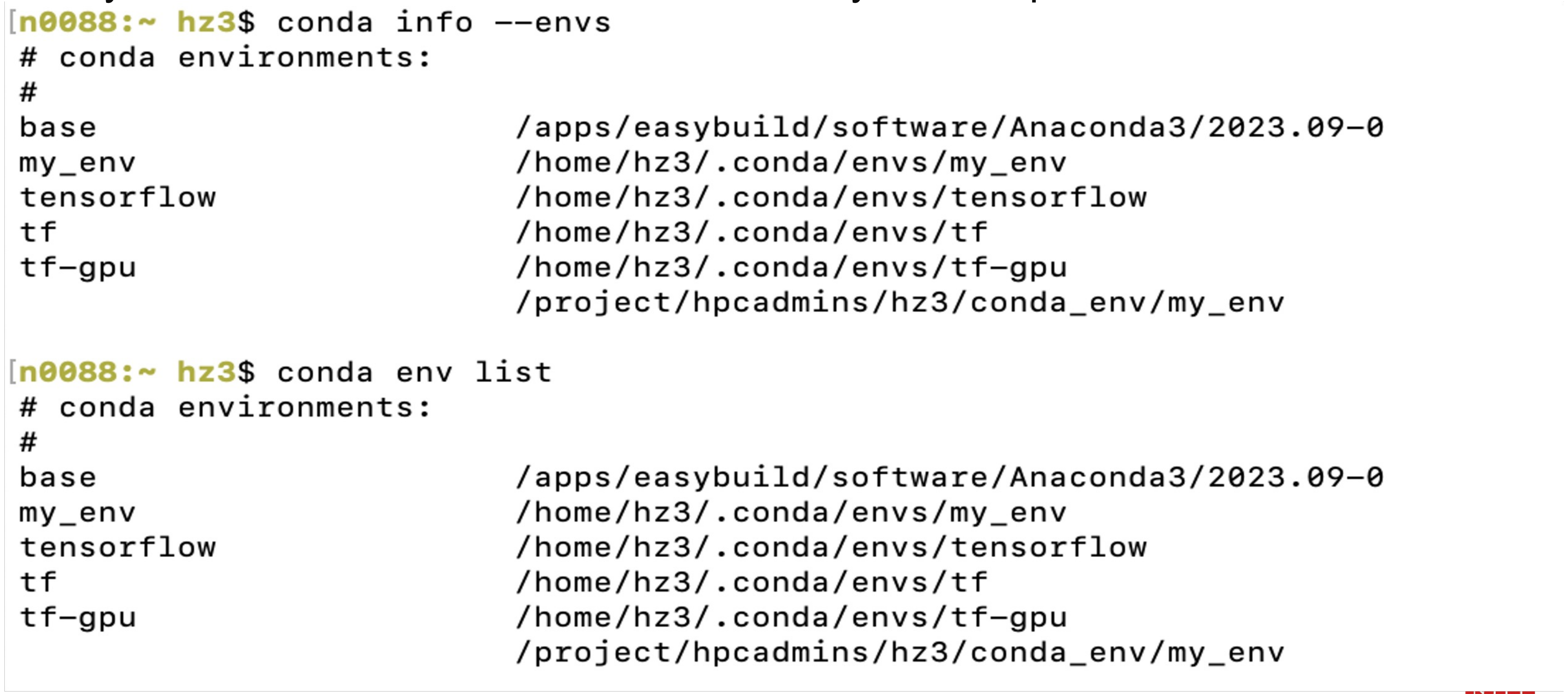
```
$ conda deactivate
```

Removing a Conda environment

```
$ conda env remove -n my_env
```

List Anaconda virtual environments

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



```
[n0088:~ hz3$ conda info --envs
 # conda environments:
 #
 base                     /apps/easybuild/software/Anaconda3/2023.09-0
 my_env                   /home/hz3/.conda/envs/my_env
 tensorflow               /home/hz3/.conda/envs/tensorflow
 tf                       /home/hz3/.conda/envs/tf
 tf-gpu                   /home/hz3/.conda/envs/tf-gpu
                          /project/hpcadmins/hz3/conda_env/my_env

[n0088:~ hz3$ conda env list
 # conda environments:
 #
 base                     /apps/easybuild/software/Anaconda3/2023.09-0
 my_env                   /home/hz3/.conda/envs/my_env
 tensorflow               /home/hz3/.conda/envs/tensorflow
 tf                       /home/hz3/.conda/envs/tf
 tf-gpu                   /home/hz3/.conda/envs/tf-gpu
                          /project/hpcadmins/hz3/conda_env/my_env
```

Conda on HPC

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

What is a channel in Conda

A channel is the location where packages are stored remotely.

When you install Conda for the first time, it comes with a channel called default.

```
$ conda config --show channels
```

You can add a channel to the list of channels using the conda config --add channels

```
$ conda config --add channels conda-forge
```

More on Channels later …

Configuring Conda channels

How can I see conda's configuration values?

```
$ conda config --help

$ conda config --show
$ conda config --show channels
        channels:
                - defaults
$conda config --describe channels
$conda config --add channels conda-forge
```

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

```
$conda config --append channels conda-forge
```

This would add the conda-forge to the end of the channel list, giving it the lowest priority.

Conda on HPC

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

Check Conda packages

List All Installed Packages:

- **conda list**
- This command 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**

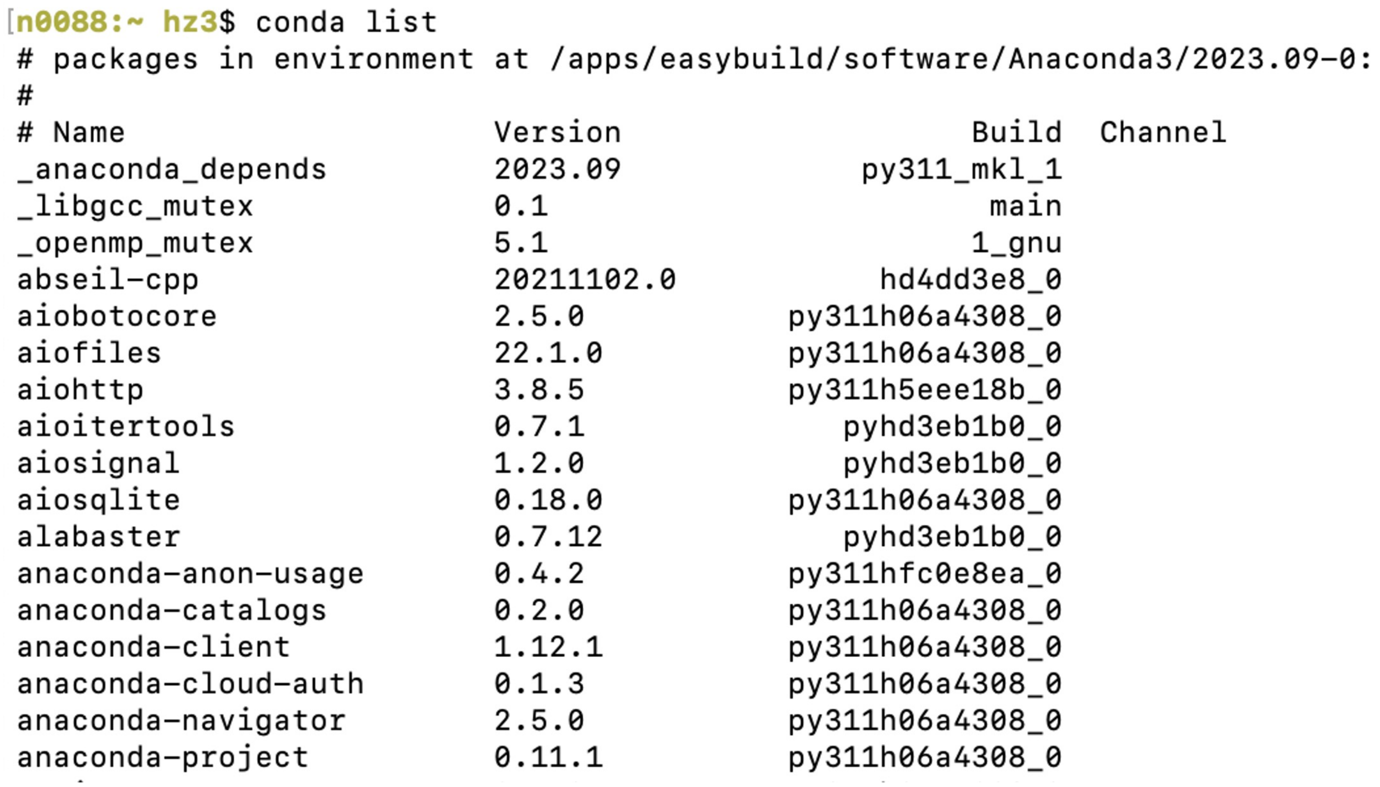
Search for a Package:

- **conda search package_name**
- This command searches for a package across all channels in Conda.

Check for Specific Package Installation:

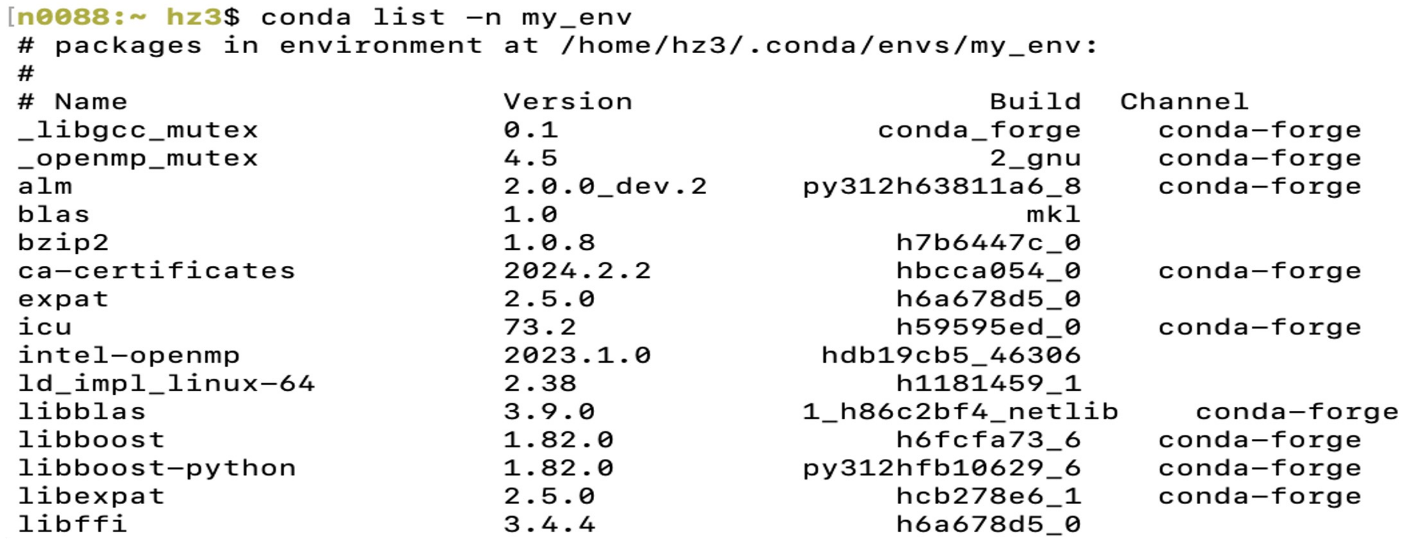
- **conda list | grep package_name**
- This command filters the list of installed packages to show only the entries related to package_name.

List packages in all environments



```
[n0088:~ hz3$ conda list
# packages in environment at /apps/easybuild/software/Anaconda3/2023.09-0:
#
# Name                    Version                   Build  Channel
_anaconda_depends         2023.09             py311_mkl_1
_libgcc_mutex             0.1                        main
_openmp_mutex             5.1                       1_gnu
abseil-cpp                20211102.0           hd4dd3e8_0
aiobotocore               2.5.0           py311h06a4308_0
aiofiles                  22.1.0          py311h06a4308_0
aiohttp                   3.8.5           py311h5eee18b_0
aioitertools              0.7.1              pyhd3eb1b0_0
aiosignal                 1.2.0              pyhd3eb1b0_0
aiosqlite                 0.18.0          py311h06a4308_0
alabaster                 0.7.12             pyhd3eb1b0_0
anaconda-anon-usage       0.4.2           py311hfc0e8ea_0
anaconda-catalogs         0.2.0           py311h06a4308_0
anaconda-client           1.12.1          py311h06a4308_0
anaconda-cloud-auth       0.1.3           py311h06a4308_0
anaconda-navigator        2.5.0           py311h06a4308_0
anaconda-project          0.11.1          py311h06a4308_0
```

List packages in an environment



```
[n0088:~ hz3$ conda list -n my_env
# packages in environment at /home/hz3/.conda/envs/my_env:
#
# Name                    Version                   Build  Channel
_libgcc_mutex             0.1                 conda_forge    conda-forge
_openmp_mutex             4.5                       2_gnu    conda-forge
alm                       2.0.0_dev.2     py312h63811a6_8    conda-forge
blas                      1.0                         mkl
bzip2                     1.0.8                h7b6447c_0
ca-certificates           2024.2.2             hbcca054_0    conda-forge
expat                     2.5.0                h6a678d5_0
icu                       73.2                 h59595ed_0    conda-forge
intel-openmp              2023.1.0         hdb19cb5_46306
ld_impl_linux-64          2.38                 h1181459_1
libblas                   3.9.0           1_h86c2bf4_netlib    conda-forge
libboost                  1.82.0               h6fcfa73_6    conda-forge
libboost-python           1.82.0          py312hfb10629_6    conda-forge
libexpat                  2.5.0                hcb278e6_1    conda-forge
libffi                    3.4.4                h6a678d5_0
```

List the installed packages for the present environment

(myenv) $ conda list

Installing Conda packages

1. Entering a Conda environment

   ```
   $ conda activate my_env
   (my_env) $: conda install scipy=1.6 --channel conda-forge
   ```

2. Create an environment called my_biowork-env and install blast from the bioconda channel:

   ```
   $ conda create --name my_biowork-env blast --channel bioconda
   ```

3. The name flag can be used to specify the environment in which we install the package

   ```
   $ conda install -n my_env scipy
   ```

4.
   ```
   $ conda install conda-forge::tensorflow  --prefix /project/$GROUP/$USER/my_env
   ```

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
- Conceived as 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 Mamba Anaconda3

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

Conda on HPC

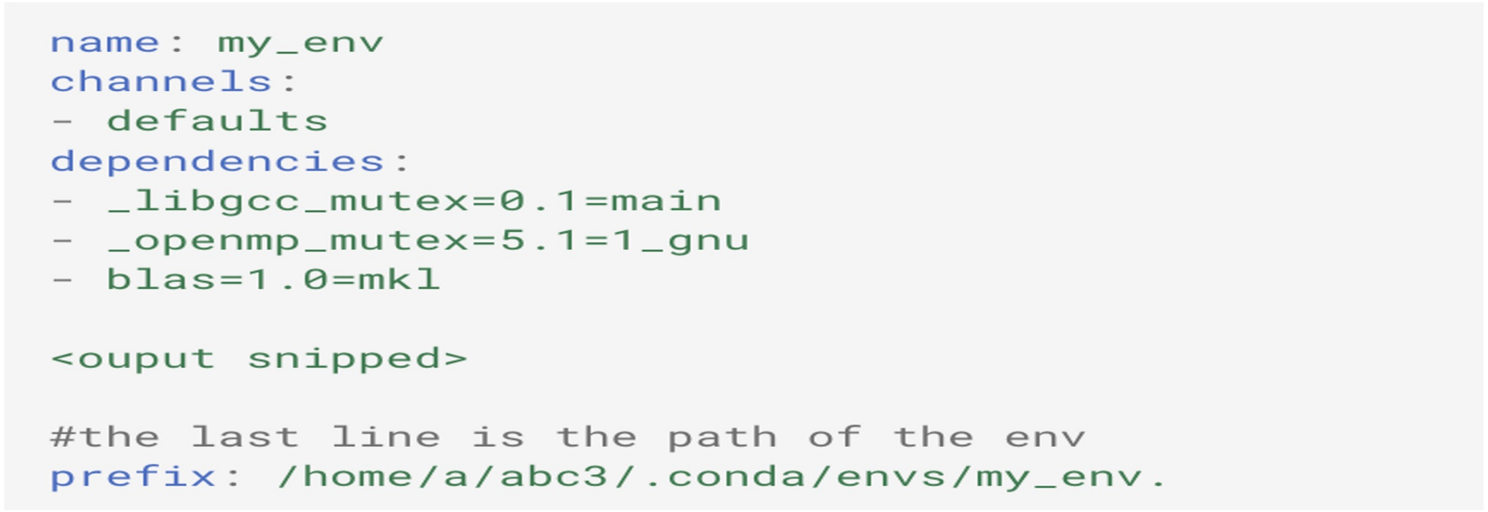
- Introduction to Conda
- Conda environment
- Conda channels
- Conda packages
- **Sharing environments**

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.
```

Importing an environment on a new machine

On the new machine,

1. First load Anaconda and initialize conda as before.
2. Then, 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

If you want to import the conda environment to a different location, use the **--prefix** or **-p** option

```
$ 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.

You need to 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 /apps/easybuild/software/Anaconda3/2023.09-0
* /project/$GROUP/$USER/conda_env/my_env
```

Updating a Conda environment

When to update your conda environment?

- One of your core dependencies just released a new version
- You need an additional package for data analysis (add a new dependency).
- You have found a better visualization package and no longer need to old visualization package

Update the contents of your environment.yml file and run the following command:

```
$ conda env update --file environment.yml  --prune
```

--prune option tells Conda to remove any dependencies that are no longer required from the environment

Best practices

Use interactive sessions on compute node

Use an interactive session on a compute node to install software with Conda to avoid slowing down the login node

```
$ srun -p general -n 1 --qos=standard --account=PI_ucid --mem-per-cpu=2G --time=59:00 --pty bash
```

#modify srun options as desired

Use /project directory with large quotas

Use /project directory other than the home directory for conda environments and packages. Using your home directory can fill its limited space.

Managing Conda Cache and changing the default caching behavior

Avoid installing packages into your `base` Conda environment

Managing Conda Cache

Default location for Conda cache files is the user's home directory. This can be changed by setting the **pkgs_dirs** entry in the **.condarc** file or setting the **CONDA_PKGS_DIRS** environment variable.

```
$ conda info
 package cache : /apps/easybuild/software/Anaconda3/2023.09-0/pkgs
                 /home/$USER/.conda/pkgs
```

The **package cache** entry will display the current package cache directories. Editing/creating the **pkgs_dirs** entry in the **.condarc** file will change the cache directory:

```
pkgs_dirs:
  -   /path/to/desired/cache/directory
```

You can also do one of the following:

- run command "**conda config --add pkgs_dirs /project/$GROUP/$USER/conda_env/pkgs_dirs**"
- setting the **CONDA_PKGS_DIRS** environment variable:

```
export CONDA_PKGS_DIRS=/path/to/desired/cache/directory
```

Use "**conda info**" to confirm the change

To see the many additional configuration options, check the official .condarc user guide here

Pip vs Conda

If your package exists on PyPI and Anaconda, how do you decide which to install from?

- **Always favor conda over pip**
- Conda (+Pip): Conda wherever possible; Pip only when necessary
- conda packages are pre-compiled and their dependencies are automatically handled.
- pip installs will often download a binary wheel (pre-compiled), the user frequently needs to take action to satisfy the dependencies.
- One disadvantage of conda packages is that they tend to lag behind pip packages in terms of versioning.

Pip installs in a Conda environment

Recommend

- Use conda environments for isolation
- Use pip only after conda, **avoid** installing conda packages after doing pip installs within a Conda environment.

```
$ conda create --name my_env pandas
$ conda activate my_env
(my_env)$ python -m pip install --user multiregex
```

- Recreate the entire environment if changes are needed after pip packages have been installed
- Use the --no-cache-dir option for pip installation commands to prevent pip filling your home directory with cached data
- Refer to Conda guide for using pip in a Conda environment

Common Python libraries for scientific computing

Library	Key Features	Common Use Cases
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

Example - install tensorflow-gpu

```
$conda create --name tensorflow python=3.9
$conda activate tensorflow
$conda install -c anaconda tensorflow-gpu numpy=1.21.6
```

Simple TensorFlow test program to make sure the virtual env can access a GPU.

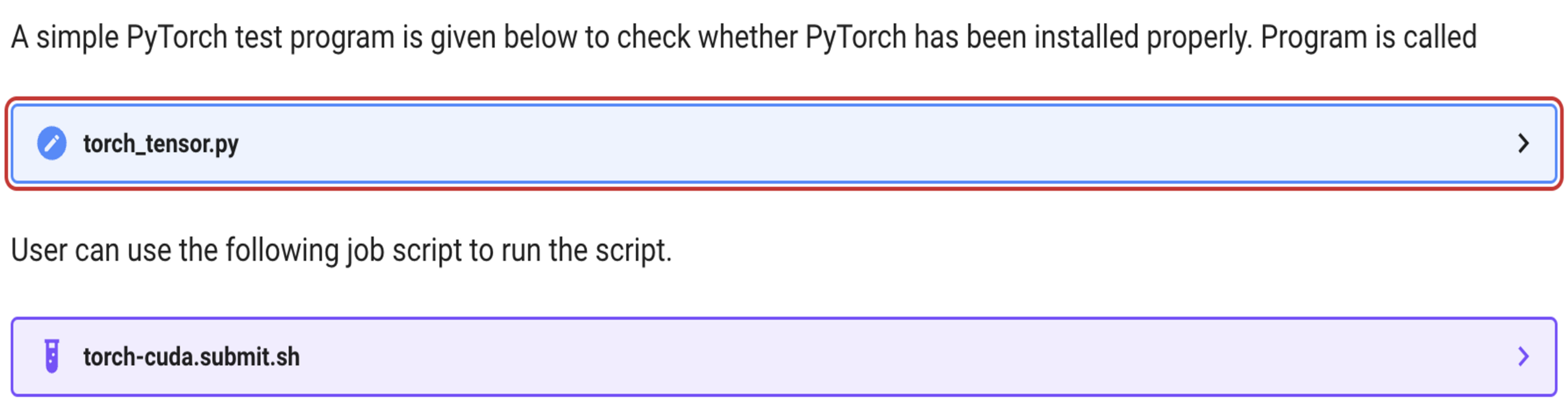
tf.gpu.test.py

Slurm script to submit the job

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

Example - Install PyTorch with GPU

```
$conda create --name torch-cuda python=3.7
$conda activate torch-cuda
$conda install -c "nvidia/label/cuda-11.7.0" cuda-toolkit
$conda install -c pytorch -c nvidia pytorch torchvision torchaudio pytorch-cuda=11.7
```



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

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

Questions?