Python on Frontier

Jamie Finney
HPC Software Support Engineer
Oak Ridge Leadership Computing Facility – OLCF
Oak Ridge National Laboratory – ORNL
August 24, 2023
Overview

- What To Expect
- Virtual Environments Overview
- Cray-Python
- Miniconda
- Best Practices
What To Expect On Frontier

• X86 Architecture
  – Easier to install from pre-compiled binaries
  – Source installs are “easier”
  – Works with conda/mamba and pip

• Should work better with Slurm, similar to Andes

• GPU workflow is now the biggest hurdle with the switch to AMD
  – Won’t be covered today

• No OLCF-provided Anaconda module
Virtual Environments

- Overview and Options
Virtual Environments

• What are they?
  – Isolated directory trees that help manage various packages or different versions of Python

• Why are they beneficial?
  – Dependencies of one package might clash with dependencies of another
  – Allows installation of new packages without modifying the “base” environment
  – Unique environments can be used on a per-project basis

• We will only discuss Python 3 today
Virtual Environment Options

• Native option: venv
  – Included with every installation
  – Extends managing your current installation

• Anaconda distribution: conda
  – Highly customizable environments
  – Large repository of supported packages
  – Not only for Python

• Pyenv, pipenv, Poetry, etc
  – Won’t be covered today
Options on Frontier

Two main options:

1. Use the cray-python module
   - Supports venv syntax
   - Comes with pre-installed packages like numpy, scipy, mpi4py tuned for Cray machines

2. Install your own Miniconda
   - Supports conda syntax
   - Similar workflow to what is used on Andes and Summit
## Comparison

### Cray-Python module

**Pros:**
- Works out of the box
- No installation needed
- Pre-installed libraries tuned for Cray

**Cons:**
- Extremely minimal
- Highly dependent on pip
- Restricted to version of module
- Can’t switch between different Python versions easily using venv

### User-Installed Miniconda

**Pros:**
- Can manage multiple Python versions
- “Easy” to install dependencies based on your current environment
- Highly customizable
- Similar workflow across OLCF systems

**Cons:**
- Must install yourself
- Can clash with loaded modules if not careful
- Highly dependent on pre-compiled libraries (can still use pip)
Cray-Python Module

- Overview and Examples
Cray-python module

- Availability:
  ```bash
  [jmfinney@login02.frontier py_on_frontier]$ module -t av cray-python
  /opt/cray/pe/lmod/modulefiles/core:
  cray-python/3.9.12.1
  cray-python/3.9.13.1
  cray-python/3.10.10
  ```

- Module version matches Python version:
  ```bash
  [jmfinney@login02.frontier py_on_frontier]$ module load cray-python
  [jmfinney@login02.frontier py_on_frontier]$ python3 -V
  Python 3.9.13
  [jmfinney@login02.frontier py_on_frontier]$ module swap cray-python/3.9.13.1 cray-python/3.10.10
  The following have been reloaded with a version change:
  1) cray-python/3.9.13.1 => cray-python/3.10.10
  [jmfinney@login02.frontier py_on_frontier]$ python3 -V
  Python 3.10.10
  ```
Cray-python module: Using Virtual Environments

To create a virtual environment:

```
$ python3 -m venv /path/to/my_env
```

This creates a set of directories at the specified location, which will contain everything unique to that virtual environment.
Cray-python module: Using Virtual Environments

To activate the environment:

$ source /path/to/my_env/bin/activate

To deactivate:

$ deactivate
Using a shebang line:

```bash
#!/path/to/my_env/bin/python3
"""
my_python_app
"""
...
Cray-python module: Installing new packages

```bash
## In general:
$ pip install <pkg>

## From source installs:
$ pip install --no-binary=<pkg>

## Using environment variables:
$ CC=gcc pip install <pkg>

## Ignore cache directory:
$ pip install --no-cache-dir <pkg>

## Upgrading package:
$ pip install --upgrade <pkg>

## In general, safer to do:
$ /path/to/my_env/bin/python3 -m pip install <pkg>
```
Cray-python module: Workflow Example

```bash
## Load cray-python module (default version), swap to GNU
$ module load cray-python
$ module swap PrgEnv-crty PrgEnv-gnu

## Create a directory to hold my environments
$ mkdir $HOME/my_envs

## Create a virtual environment called "mpi4py_env" in my environments folder
$ python3 -m venv $HOME/my_envs/mpi4py_env

## Activate the virtual environment
$ source $HOME/my_envs/mpi4py_env/bin/activate

## Install mpi4py
( mpi4py_env)$ MPICC="cc -shared" pip install --no-cache-dir --no-binary=mpi4py mpi4py
```
Cray-python module: Workflow Example

[jmfinney@login02.frontier py_on_frontier]$ module load cray-python
[jmfinney@login02.frontier py_on_frontier]$ module swap PrgEnv-cray/8.3.3 PrgEnv-gnu
Lmod is automatically replacing "cce/15.0.0" with "gcc/12.2.0".

Due to MODULEPATH changes, the following have been reloaded:
  1) cray-mpich/8.1.23   2) darshan-runtime/3.4.0

[jmfinney@login02.frontier py_on_frontier]$ python3 -m venv ./mpi4py_env
[jmfinney@login02.frontier py_on_frontier]$ . ./mpi4py_env/bin/activate

( mpi4py_env ) [jmfinney@login02.frontier py_on_frontier]$ MPICC="cc -shared" pip install mpi4py
Collecting mpi4py
  Using cached mpi4py-3.1.4-cp310-cp310-linux_x86_64.whl
Installing collected packages: mpi4py
Successfully installed mpi4py-3.1.4
Cray-python module: Workflow Example

```
(mpi4py_env) [jmfinney@login02.frontier py_on_frontier]$ pip list
Package         Version
---------------    -------
mpi4py           3.1.4
pip              22.3.1
setuptools       65.5.0

[notice] A new release of pip available: 22.3.1 -> 23.2.1
[notice] To update, run: pip install --upgrade pip
```
Cray-python module: Workflow Example

```
(mpi4py_env) [jmfinney@login02.frontier py_on_frontier]$ salloc -A stf243 -N 1 -t 00:05:00
salloc: Pending job allocation 1414442
salloc: job 1414442 queued and waiting for resources
salloc: job 1414442 has been allocated resources
salloc: Granted job allocation 1414442
salloc: Waiting for resource configuration
salloc: Nodes frontier08190 are ready for job

jmfinney@frontier08190:~/frontier/projects/py_on_frontier> srun --pty python3
Python 3.10.10 (main, Apr 14 2023, 19:14:32) [GCC 9.3.0 20200312 (Cray Inc.)] on linux
Type "help", "copyright", "credits" or "license" for more information.

>>> from mpi4py import MPI
>>> MPI.Get_library_version()
'MPI VERSION   : CRAY MPICH version 8.1.23.14 (ANL base 3.4a2)\nMPI BUILD INFO : Tue Nov 29 12:34 2022 (git hash 210ae8b)\n'
```
Cray-python module: Workflow Example

```bash
#!/bin/bash
#SBATCH --A stf243
#SBATCH --J hello_py
#SBATCH --o %x-%j.out
#SBATCH --t 0:05:00
#SBATCH --p batch
#SBATCH --N 1

unset SLURM_EXPORT_ENV

module load cray-python/3.9.13.1
module swap PrgEnv-cray PrgEnv-gnu
module -t list

cd /ccs/proj/stf243/jmfinney/python/
source ./my_env/bin/activate

srun python3 hello.py
```
Cray-python module: Example

```python
#!/autofs/nccs-svml_home1/jmfinney/frontier/projects/py_on_frontier/my_env/bin/python

""
Parallel Hello World
""

from mpi4py import MPI
import sys

size = MPI.COMM_WORLD.Get_size()
rank = MPI.COMM_WORLD.Get_rank()
name = MPI.Get_processor_name()

sys.stdout.write(
    "Hello, World! I am process %d of %d on %s.\n"
% (rank, size, name))
```
Cray-Python: Documentation

- See Python on OLCF Systems:
  - [https://docs.olcf.ornl.gov/software/python/index.html](https://docs.olcf.ornl.gov/software/python/index.html)

- Official Python venv documentation:
  - [https://docs.python.org/3/tutorial/venv.html](https://docs.python.org/3/tutorial/venv.html)
Miniconda

- Installation and usage
Miniconda

• What is Miniconda?
  – A free minimal installer for conda. It is a small bootstrap version of Anaconda that includes only conda, Python, the packages they both depend on, and a small number of other useful packages (like pip, zlib, and a few others).

• If your workflow better suits conda environments, you can install your own Miniconda: https://docs.conda.io/en/main/miniconda.html
  – Also, please submit a ticket to help@olcf.ornl.gov saying that conda is better for your workflow
Miniconda: Installation

$ mkdir miniconda_frontier && cd miniconda_frontier
$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
$ chmod u+x Miniconda3-latest-Linux-x86_64.sh
$ ./Miniconda3-latest-Linux-x86_64.sh -u -p ./miniconda_frontier

- **-p** specifies the prefix path for where to install miniconda
- **-u** updates any current installations at the “-p” location (not necessary if you didn’t do a “mkdir” beforehand)
Miniconda: Installation

- While running the installer you will be prompted with:

  Do you wish the installer to initialize Miniconda3 by running conda init? [yes|no]

- If “yes”, your `~/.bashrc` (or equivalent shell configuration file) will be updated with:

  ```bash
  # >>> conda initialize >>>
  # !! Contents within this block are managed by ’conda init’ !!
  .
  .
  .
  #unset __conda_setup
  # <<< conda initialize <<<
  ```
Miniconda: Installation Warning

- **Warning:** By default, this will always initialize conda upon login, which clashes with other Python installations (e.g., if you use the anaconda modules on other OLCF systems). It is **MUCH SAFER** to say “no” and to just export the PATH manually when on Frontier to avoid clashing:

  ```bash
  $ export PATH="/path/to/your/miniconda/bin:$PATH"
  ```

- **Note:** If your `.bashrc` is already modified (from other OLCF modules), then it will **NOT** modify your `.bashrc`

- **Highly recommend this (only needs to be run once):**

  ```bash
  $ conda config --set autoActivateBase false
  ```
Miniconda: Using Conda Environments

• Create an environment:

```bash
# Named env in default location
$ conda create -n my_env python=3.9,3.10,...

# Explicit path to env location
$ conda create -p /path/to/my_env python=3.9,3.10,...
```

• Activate a base environment first (if conda not started):

```bash
$ source /path/to/my_env/bin/activate
```

• Activate/Deactivate an environment (with conda started):

```bash
$ conda activate my_env
$ conda activate /path/to/my_env
$ conda deactivate
```
Miniconda: Using Conda Environments

- Install packages using default channel:

  $ conda install <package_name>

- Install package using explicit channel name (ie conda-forge):

  $ conda install -c conda_forge <package_name>
Miniconda: Installation

[jmfinney@login02.frontier py_on_frontier]$ cd miniconda/
[jmfinney@login02.frontier miniconda]$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh --2023-08-22 15:43:52-- https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.130.3, 104.16.131.3, 2606:4700::6810:8203, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.130.3|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 103219356 (98M) [application/x-sh]
Saving to: ‘Miniconda3-latest-Linux-x86_64.sh’
Miniconda3-latest-Linux-x86_64.sh 100%[==========================================] 98.44M 219MB/s in 0.4s

2023-08-22 15:43:53 (219 MB/s) - ‘Miniconda3-latest-Linux-x86_64.sh’ saved [103219356/103219356]
Miniconda: Installation

[jmfinney@login02.frontier miniconda]$ chmod u+x Miniconda3-latest-Linux-x86_64.sh
[jmfinney@login02.frontier miniconda]$ ./Miniconda3-latest-Linux-x86_64.sh -p ./conda

Welcome to Miniconda3 py311_23.5.2-0

In order to continue the installation process, please review the license agreement.
Please, press ENTER to continue

>>>=
End User License Agreement - Miniconda

Copyright 2015-2023, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:
...
Miniconda: Installation

... Last updated March 21, 2022

Do you accept the license terms? [yes|no]
[no] >>> yes

Miniconda3 will now be installed into this location:
./conda

- Press ENTER to confirm the location
- Press CTRL-C to abort the installation
- Or specify a different location below

[./conda] >>>
PREFIX=/ccs/home/jmfinney/frontier/projects/py_on_frontier/miniconda/conda
Unpacking payload ...
Installing base environment...
Downloading and Extracting Packages

Preparing transaction: done
Executing transaction: done
installation finished.
...

Miniconda: Installation

... installation finished.

WARNING:
- You currently have a PYTHONPATH environment variable set. This may cause unexpected behavior when running the Python interpreter in Miniconda3.
- For best results, please verify that your PYTHONPATH only points to directories of packages that are compatible with the Python interpreter in Miniconda3: /ccs/home/jmfinney/frontier/projects/py_on_frontier/miniconda/conda

Do you wish the installer to initialize Miniconda3 by running conda init? [yes|no]
[no] >>> no

You have chosen to not have conda modify your shell scripts at all.

To activate conda's base environment in your current shell session:

eval "$(/ccs/home/jmfinney/frontier/projects/py_on_frontier/miniconda/conda/bin/conda shell.YOUR_SHELL_NAME hook)"

To install conda's shell functions for easier access, first activate, then:

conda init

If you'd prefer that conda's base environment not be activated on startup, set the auto_activate_base parameter to false:

conda config --set auto_activate_base false

Thank you for installing Miniconda3!
Miniconda: Example

```bash
[jemfinney@login02.frontier miniconda]$ . ./conda/bin/activate
(base) [jemfinney@login02.frontier miniconda]$ conda list
# packages in environment at /ccs/home/jemfinney/frontier/projects/py_on_frontier/miniconda/conda:
#
<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Build</th>
<th>Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>_libgcc_mutex</td>
<td>0.1</td>
<td>main</td>
<td></td>
</tr>
<tr>
<td>_openmp_mutex</td>
<td>5.1</td>
<td>1_gnu</td>
<td></td>
</tr>
<tr>
<td>boltons</td>
<td>23.0.0</td>
<td>py311h06a4308_0</td>
<td></td>
</tr>
<tr>
<td>brotli_py</td>
<td>0.7.0</td>
<td>py311h5e6e18b_1002</td>
<td></td>
</tr>
<tr>
<td>bzip2</td>
<td>1.0.8</td>
<td>h7b5447c_0</td>
<td></td>
</tr>
<tr>
<td>c-ares</td>
<td>1.19.0</td>
<td>h5ee18b_0</td>
<td></td>
</tr>
<tr>
<td>ca-certificates</td>
<td>2023.05.30</td>
<td>h06a4308_0</td>
<td></td>
</tr>
<tr>
<td>certifi</td>
<td>2023.5.7</td>
<td>py311h06a4308_0</td>
<td></td>
</tr>
<tr>
<td>cffi</td>
<td>1.15.1</td>
<td>py311h5e6e18b_3</td>
<td></td>
</tr>
<tr>
<td>charset-normalizer</td>
<td>2.0.4</td>
<td>pyhd3eb1b0_0</td>
<td></td>
</tr>
<tr>
<td>conda</td>
<td>23.5.2</td>
<td>py311h06a4308_0</td>
<td></td>
</tr>
<tr>
<td>conda-content-trust</td>
<td>0.1.3</td>
<td>py311h06a4308_0</td>
<td></td>
</tr>
<tr>
<td>conda-libmamba-solver</td>
<td>23.5.0</td>
<td>py311h06a4308_0</td>
<td></td>
</tr>
<tr>
<td>conda-package-handling</td>
<td>2.1.0</td>
<td>py311h06a4308_0</td>
<td></td>
</tr>
<tr>
<td>conda-package-streaming</td>
<td>0.8.0</td>
<td>py311h06a4308_0</td>
<td></td>
</tr>
<tr>
<td>cryptography</td>
<td>39.0.1</td>
<td>py311h9cele76_2</td>
<td></td>
</tr>
</tbody>
</table>
```

Miniconda: Example

```
(base) [jmfinney@login02.frontier miniconda]$ conda install mpi4py
Retrieving notices: ...working... done
Collecting package metadata (current_repodata.json): done
Solving environment: done
## Package Plan ##

environment location: /ccs/home/jmfinney/frontier/projects/py_on_frontier/miniconda/conda

added / updated specs:
  - mpi4py

The following packages will be downloaded:

<table>
<thead>
<tr>
<th>package</th>
<th>build</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>certifi-2023.7.22</td>
<td>py311h06a4308_0</td>
<td>154 KB</td>
</tr>
<tr>
<td>conda-23.7.3</td>
<td>py311h06a4308_0</td>
<td>1.3 MB</td>
</tr>
<tr>
<td>libfortran-ng-7.5.0</td>
<td>ha8ba4b0_17</td>
<td>22 KB</td>
</tr>
<tr>
<td>libfortran4-7.5.0</td>
<td>ha8ba4b0_17</td>
<td>995 KB</td>
</tr>
<tr>
<td>mpi-1.0</td>
<td>mpich</td>
<td>13 KB</td>
</tr>
<tr>
<td>mpi4py-3.1.4</td>
<td>py311hfc96b8b0_0</td>
<td>540 KB</td>
</tr>
<tr>
<td>mpich-3.3.2</td>
<td>hc856adb_0</td>
<td>3.8 MB</td>
</tr>
<tr>
<td>openssl-3.0.10</td>
<td>h7f8727e_1</td>
<td>5.2 MB</td>
</tr>
</tbody>
</table>

Total: 12.1 MB
```
Miniconda: Example

The following NEW packages will be INSTALLED:

- libgfortran-ng @ pkgs/main/linux-64::libgfortran-ng-7.5.0-ha8ba4b0_17
- libgfortran4 @ pkgs/main/linux-64::libgfortran4-7.5.0-ha8ba4b0_17
- mpi @ pkgs/main/linux-64::mpi-1.0-mpich
- mpi4py @ pkgs/main/linux-64::mpi4py-3.1.4-py311hfc96bbd_0
- mpich @ pkgs/main/linux-64::mpich-3.3.2-hc856adb_0

The following packages will be UPDATED:

- certifi 2023.5.7-py311h06a4308_0 --> 2023.7.22-py311h06a4308_0
- conda 23.5.2-py311h06a4308_0 --> 23.7.3-py311h06a4308_0
- openssl 3.0.9-h7f8727e_0 --> 3.0.10-h7f8727e_1

Proceed ([y]/[n])? y

Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Miniconda: Example

(base) [jmfinney@login02.frontier miniconda]$ conda deactivate
[jmfinney@login02.frontier miniconda]$ sbatch launch.sh
Submitted batch job 1416116

[jmfinney@login02.frontier miniconda]$ cat hello_py-1416028.out
Lmod is automatically replacing "cce/15.0.0" with "gcc/12.2.0".
Lmod is automatically replacing "PrgEnv-cray/8.3.3" with "PrgEnv-gnu/8.3.3".

Due to MODULEPATH changes, the following have been reloaded:
  1) cray-mpich/8.1.23  2) darshan-runtime/3.4.0

Hello, World! I am process 0 of 1 on frontier01910.
Hello, World! I am process 0 of 1 on frontier01920.
Hello, World! I am process 0 of 1 on frontier01918.
Hello, World! I am process 0 of 1 on frontier01915.
Miniconda: Example

```bash
#!/bin/bash
#SBATCH --A stf243
#SBATCH --J hello_py
#SBATCH --o %x-%j.out
#SBATCH --t 0:05:00
#SBATCH --p batch
#SBATCH --N 4

unset SLURM_EXPORT_ENV

module load PrgEnv-gnu

cd /ccs/home/jmfinney/frontier/projects/py_on_frontier/miniconda
source conda/bin/activate
	srun python hello.py
```
Miniconda: Documentation Links

• See our Conda Basics guide with a quick-reference list here:
  – https://docs.olcf.ornl.gov/software/python/conda_basics.html#condaquick

• Conda’s official user guide:
Best Practices
Best Practices

• Most default environment locations are at $HOME on NFS, be careful storing things in $MEMBERWORK or $PROJWORK because it might get purged.
  – For collaboration, use “Project Home”:
    `/ccs/proj/<projid>`

• Make note of your pip cache location by running:
  – May need to clean it from time-to-time:
    `$ pip cache info`
    `$ pip cache purge`

• Similarly, clean your conda cache occasionally:
    `$ conda clean --all`

• Explicitly use “python3” instead of the “python” alias
Best Practices

• In general, most python packages assume use of GCC
  – Recommended to use PrgEnv-gnu, especially when building from source

• Deactivate virtual environments first before switching PrgEnv modules

• Deactivate virtual environments before entering batch/interactive jobs
  – Some deactivation syntax won’t work properly if entering a job already activated
  – Always better to enter any form of job with a fresh login shell and module environment

• When submitting a batch job that uses virtual environments:
  – Activate all your modules / your virtual env in the batch script.

  Use at CLI:
  ```
  $ sbatch --export=NONE submit.sl
  ```

  In batch script:
  ```
  unset SLURM_EXPORT_ENV
  ```
Best Practices

• Similar to Andes and Summit, it’s always recommended to “clone” the base environment before trying to install packages.
  
  – For venv:

    
    ```
    $ python3 -m venv /path/to/new_env --system-site-packages
    ```

  – Cloning with conda (does not really apply to Crusher/Frontier):

    
    ```
    $ conda create -n new_env --clone base
    ```
Best Practices

• To “export”/“import” your current environment:
  – For venv:

    $ python3 -m pip freeze > requirements.txt
    $ python3 -m pip install -r requirements.txt

  – For conda:

    $ conda env export > environment.yml
    $ conda env create -f environment.yml
Questions?