Summit Python Environments

Intro to Summit Webinar
1 June 2018

Presenters: Matt Belhorn

Getting Started

- RHEL7 provides python v2.7.5 as default.
- Alternates available through environment modules

```
$ module -w 80 avail python

------------ /sw/summit/modulefiles/site/linux-rhel7-ppc64le/Core ------------
   python/2.7.12    python/3.5.2 (D)

  Where:
   D:  Default Module
```

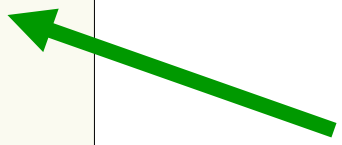
Getting Started (cont'd)

- Following PEP394, `python` refers to `python2`. Version 3.x interperters must be invoked as `python3`.
- It is recommended that scripts use version-specific shebang lines:

```
#!/usr/bin/env python3.5

#!/path/to/my/venv/python

#!/usr/bin/env python2.7
```



Conveniently runs within the venv environment when invoked this way.

Where are the extensions?

- Python environment modules will include small set of basic packages
  - `pip`, `virtualenv` (python2 only), `setuptools`, `nose`
  - Modulefiles for pip, virtualenv, setuptools, nose, and other packages included when loading python core environment module will eventually disappear.
- Additional extensions available through environment modules.
  - Available extensions affected by choice of compiler environment.
  - OS-provided `gcc/4.8.5` provides greatest support for extensions.

Where are the extensions? (Cont'd)

Duplicate packages eclipsed by currently loaded modules

```
$ module -w 80 avail py

-------------- spectrum-mpi/10.2.0.0-20180508-iuhmuvf/gcc/4.8.5 ----------------
   adios/1.11.1-py2    petsc/3.7.2-py2 (D)    py-mpi4py/2.0.0-py2
   petsc/3.6.3-py2     py-h5py/2.6.0-py2      py-mpi4py/2.0.0-py3 (D)
   petsc/3.6.4-py2     py-h5py/2.6.0-py3 (D)

--------- /sw/summit/modulefiles/site/linux-rhel7-ppc64le/gcc/4.8.5 -----------
   py-h5py/2.6.0-py2    py-numpy/1.11.2-py2        python/2.7.12
   py-h5py/2.6.0-py3    py-numpy/1.11.2-py3 (D)    python/3.5.2  (D)
   py-nose/1.3.7-py2    py-scipy/0.18.1-py2
   py-nose/1.3.7-py3    py-scipy/0.18.1-py3 (D)

------------- /sw/summit/modulefiles/site/linux-rhel7-ppc64le/Core ------------
   py-nose/1.3.7 (D)    py-setuptools/25.2.0    python/2.7.12
   py-pip/9.0.1         py-virtualenv/15.0.1    python/3.5.2
```

Plan to incorporate common basic extensions into core interpereter modules.
Load them by hand for now.

Where are the extensions? (Cont'd)

Packages built for use with python2 or python3 (either extensions or bindings) suffixed -py2 and -py3

```
$ module -w 80 avail py

-------------- spectrum-mpi/10.2.0.0-20180508-iuhmuvf/gcc/4.8.5 ---------------
   adios/1.11.1-py2    petsc/3.7.2-py2   (D)    py-mpi4py/2.0.0-py2
   petsc/3.6.3-py2     py-h5py/2.6.0-py2        py-mpi4py/2.0.0-py3 (D)
   petsc/3.6.4-py2     py-h5py/2.6.0-py3 (D)

---------- /sw/summit/modulefiles/site/linux-rhel7-ppc64le/gcc/4.8.5 ----------
   py-h5py/2.6.0-py2    py-numpy/1.11.2-py2        python/2.7.12
   py-h5py/2.6.0-py3    py-numpy/1.11.2-py3 (D)    python/3.5.2  (D)
   py-nose/1.3.7-py2    py-scipy/0.18.1-py2
   py-nose/1.3.7-py3    py-scipy/0.18.1-py3 (D)

------------- /sw/summit/modulefiles/site/linux-rhel7-ppc64le/Core ------------
   py-nose/1.3.7 (D)    py-setuptools/25.2.0    python/2.7.12
   py-pip/9.0.1         py-virtualenv/15.0.1    python/3.5.2
```

Either will add packages to the PYTHONPATH, so choose wisely and avoid mixing interpereters.

I want some more extensions!

- If it’s a very common extension that depends on external libraries and you are comfortable using the OS compiler environment, let us know and we’ll consider installing it as an environment module.
- Otherwise, consider using a virtualenv and managing your own python stack.
  - Cons: greater initial setup time
  - Pros: Your python runtime is exactly how you want it, no surprises.

Why use a virtualenv when I can `pip install --user`?

- Avoid `pip install --user` for the time being!
  - $HOME directory is shared on all OLCF machines.
  - Default $PYTHONUSERBASE is `$HOME/.local/lib/pythonX.Y/`
  - Compiled ppc64le and x86_64 packages/wheels *will* conflict.
- A unique $PYTHONUSERBASE will be added to Summit python modulefiles in the future

Creating Virtualenvs

Use a non-purged, machine/arch-specific path

```
$ MY_VENVS="/ccs/proj/<projid>/${USER}/summit/opt/venvs"
$ virtualenv "${MY_VENVS}/some_python2_venv"
$ python3 -m venv "${MY_VENVS}/some_python3_venv"
$ . $MY_VENVS/some_python3_venv/bin/activate
```

venv is built-in to the Python3 standard library: no need for virtualenv

- Load all necessary environment modules before activating venv.
- Deactivate venv before changing environment modules.
- Use LMOD collections to save/restore environment modules.
- Must build some packages from source against external libraries.
- Don't mix envmod python extensions with a venv. Put everything you need into the venv so it can be truly stand-alone.

What about Anaconda?

- OLCF will provide Anaconda through environment modules soon.
- In the meantime, it's easy to deploy your own Anaconda suite:

```
$ wget https://repo.continuum.io/archive/Anaconda2-5.2.0-Linux-ppc64le.sh
$ chmod a+x Anaconda2-5.2.0-Linux-ppc64le.sh
$ PREFIX="/ccs/proj/<projid>/${USER}/summit/opt/anaconda"
$ ./Anaconda2-5.2.0-Linux-ppc64le.sh -b -p "${PREFIX}"
$ export PATH="${PREFIX}/bin:${PATH}"
$ pip install -v --no-binary mpi4py mpi4py
```

Must rebuild some packages against system libraries. Notably compiled MPI and CUDA packages. See package documentation for build options.

Anaconda's ppc64le release does not necessarily contain all the extensions available in the x86_64 release… yet.

That’s a start, now what’s next?

- More information on Python at the OLCF is available in our previous (and future) user meeting events.

  For specific problems, please send us your questions and comments regarding the python environment to `help@olcf.ornl.gov`.

  We’re happy to help get your python applications up and running on Summit.