Ryan Abernathey

Custom Conda Environments on Clusters

A problem that lots of us have to deal with is how to run our python code on a cluster (e.g. yellowstone, discover, habanero, etc.). On our local machine or on sverdrup, we know how to manage packages using conda and pip. But most clusters do not have conda available, and the system or module-based python distribution probably doesn't have all the packages we need (e.g. xarray, dask, etc.). How can we get around these limitations to get a fully functional, flexible python environment on any cluster?

Here is my solution, which I have used succesfully on habanero and pleiades. I assume something similar will work on most clusters.

Step 1: Install miniconda in user space.

Miniconda is a mini version of Anaconda that includes just conda and its dependencies. It is a very small download. If you want to use python 3 (recommended) you can call

```
wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh -O
miniconda.sh
```

or for python 2.7

```
wget https://repo.continuum.io/miniconda/Miniconda2-latest-Linux-x86_64.sh -O
miniconda.sh
```

Step 2: Run Miniconda

Now you actually run miniconda to install the package manager. The trick is to specify the install directory within your home directory, rather in the default system-wide installation (which you won't have permissions to do). You then have to add this directory to your path.

```
bash miniconda.sh -b -p $HOME/miniconda
export PATH="$HOME/miniconda/bin:$PATH"
```

Step 3: Create a custom conda environment specification

You now have to define what packages you actually want to install. A good way to do this is with a custom conda environment file. The contents of this file will differ for each project. Below is the environment.yml that I use for my xmitgcm project.

```
name: xmitgcm
dependencies:
 - numpy
 - scipy
 - xarray
 - netcdf4
 - dask
 - jupyter
 - matplotlib
 - pip:
   - pytest
   - xmitgcm
```

Create a similar file for your project and save it as `environment.yml`. You should chose a value of `name` that makes sense for your project.

Step 4: Create the conda environment

You should now be able to run the following command

```
conda env create --file environmen.yml
```

This will download and install all the packages and their dependencies.

Step 5: Activate The Environment

The environment you created needs to be activated before you can actually use it. To do this, you call

```
source activate xmitgcm
```

(You replace `xmitgcm` with whatever name you picked in your environment.yml file.) This step needs to be repeated whenever you want to use the environment (i.e. every time you launch an interactive job or call python from within a batch job).

Step 6: Use Python!

You can now call `ipython` on the command line or launch a jupyter notebook. On most clusters, this should be done from a compute node, rather than the head node. Connecting with your notebook running on a cluster is a bit complicated, but it can definitely be done. That will be the topic of my next post.