INSTALLING AND RUNNING PYTHON PACKAGES ON RCAC COMMUNITY CLUSTERS

Feb 2, 2023

Amiya K Maji

Lead Computational Scientist
Rosen Center for Advanced Computing (RCAC)
Purdue University
Contents

- Fun with Python
- System Python vs. Anaconda module
- Challenges for installing Python packages
- Conda environments
- conda-env-mod
- Install cartopy
- Installing packages in a group shared directory
- Miniconda
- Troubleshooting
- Questions
Fun with Python

MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.
**System Python vs. Anaconda module**

- Default Python is located at `/bin/python (v2.7.5)
  - No scientific packages are pre-installed
- Always load the anaconda module
  - `module load anaconda`
- Stick to a Python/Anaconda version
  - `module load anaconda/2020.11-py38`
- Challenges for installing additional packages
  - Insufficient permission
  - Mismatched dependencies for various packages
  - Refer Slide 3 for a clear picture.
Installing Python packages

- Use **conda environments** (for additional packages)
- Use **conda install** (in your personal environment)
- Keep it simple and self-contained
- It is important to keep track of what you installed with conda/pip
- Run **conda list** to see what packages are available
- **Shortcomings of conda environments**
  - Using installed packages is painful
  - `source activate mypackage` does not work in tcsh
  - **NEVER RUN** `conda init`
  - `conda activate` followed by `conda deactivate` can destroy your environment.
**conda-env-mod: Simplifying package installation**

- Automating environment creation and configuration reduces mistakes
- Module files enable sharing of conda environments
- Automatic kernel creation allows environments to be used in Jupyter notebooks

```
conda-env-mod
create myenv

- conda create
- create module
- create kernel

module load

- pip install
- conda install
```
conda-env-mod: Features

- **Run** conda-env-mod --help
- **create**
  - Create a minimal anaconda environment
  - Python must match with base Python
- **delete**
  - Delete an existing environment
- **module**
  - Create/update module file for an existing environment
- **kernel**
  - Create Jupyter kernel for an existing environment
  - The environment must have ipython and ipykernel installed
- **Let’s install some packages!**
Ex. 1: Install cartopy using Conda

- conda-env-mod create -n cartopy
- Answer the prompts
- Note down the instructions for loading the cartopy environment
- Load necessary modules
- conda install cartopy
- which python
- conda list
- Let’s try to load cartopy
- What went wrong?
  - !@$%^&*
Test cartopy

- Use the Python inside the environment
  - conda-env-mod module -n cartopy --local-python
- Now try to import cartopy
  - Success!!!
- Run some more examples
Install packages with pip

- List which modules are loaded
- `pip install pipdeptree`
- `pipdeptree`
- `pipdeptree --graph-output png`
- `pip install graphviz`
- `pipdeptree --graph-output png > dep.png`
- `display dep.png`
Ex. 2: Install cartopy for your research group

- **Motivations**
  - Share a single lab-wide installation
  - Installations in $HOME consume space

- `conda-env-mod create -p
  /depot/mylab/apps/cartopy -m
  /depot/mylab/etc/modules --local-python`

- **Load the modules**
- `conda install cartopy`
- `conda list`
- `which python`
- **Run example codes**
Ex. 3: Use cartopy in Jupyter Notebook

- Use the **--jupyter** option in `conda-env-mod`
  - `conda-env-mod` will install IPython and IPyKernel in the environment
  - Create a Kernel definition that is visible from JupyterHub
- `conda-env-mod create -n cartopy --jupyter`
- `module load ...
- `conda install cartopy`
- Open JupyterHub and select the cartopy Kernel
Working with Miniconda

- When to use miniconda
  - You want a Python version that is not available as module
  - You want a Python that is isolated from central installations

- Download miniconda from

- Install

- Set `PATH` to miniconda installation

- Advantages
  - No need to use the anaconda module
  - Totally isolated
  - You can install any Python version that you want

- Disadvantage
  - You must manage your own installation
Python IDEs

- Spyder
- Pycharm
- Jupyter

- Spyder is already installed with the anaconda module
  - Or run `conda install spyder` in the environment

- You can install Pycharm in your home
Caveats

- Do not install packages with `pip install --user`
- Do not mix channels, create separate environments instead
- Watch out for dependencies across packages
- Watch for disk usage in your home directory
  - `myquota`
- Do not load Python/Acona in `~/.bashrc`
- Do not use `conda init`
Troubleshooting

● Always be mindful of your runtime environment
  ○ module list
  ○ echo $PYTHONPATH
  ○ echo $PATH
  ○ echo $LD_LIBRARY_PATH

● Some packages may need additional libraries.
  ○ Load appropriate modules

● When in doubt, clean up directories where Python installs packages
  mv ~/.conda ~/.conda.bak
  mv ~/.local ~/.local.bak
  mv ~/.cache ~/.cache.bak

● Other configuration locations
  ~/.jupyter  ~/.ipython  ~/.config

● Read the user guide
  ○ https://www.rcac.purdue.edu/knowledge/scholar/run/examples/apps/python/packages
QUESTIONS