Collaborative Science with Conda and Binstar

Conda

- Cross platform package manager built by Continuum Analytics
- Goes beyond pip (and friends) capability
- Endorsed by Python Packaging Authority (PyPA)

What is a package manager?

- “collection of software tools to automate the process of installing, upgrading, configuring, and removing software packages” - wikipedia
- In practical terms, “I am a researcher, and I need to import numpy or cartopy. How do I do that?”

Problems w/ traditional Python Packaging

- Stackoverflow Q & A: “Differences between distribute, distutils, setuptools and distutils2?”
- “Python packaging/installation has way too many alternatives with no clear guidance from the community.” -Sabuncu
- “I love Python, but the state of Python packaging is nothing less than hellish!” -Zearin

pip

- Works well for Python
- Not great if you are linking against C and Fortran libraries (e.g., HDF5)

Conda to the rescue

- Python agnostic package manager
- Cross-platform
- No admin privileges required
- Smart dependency management
- Easy to work w/ different versions of packages (e.g., numpy 1.7 vs. 1.9)
- Free and available at Continuum Analytics

Some conda definitions. . .

Conda “packages”

- Binary tarballs containing system-level libraries, Python modules, executable programs
- Examples: numpy, matplotlib, ipython, libnetcdf, etc.
- Can also build packages for distribution via binstar channels

Conda “environments”

- conda environment is a collection of packages
- Simply a directory on the file system containing conda packages
- Environments nicely compartmentalized
- Easy to set up environments
- Easy to invoke and switch between environments

Conda “channels”

- conda packages originate from “channels”
- There are default channels for most standard packages
- Add custom channels to find special packages
- You can become your own channel binstar
- Examine channel list in .condarc

Working with conda from the command line

The conda command

- Primary interface for managing Python packages

Asking conda for help

- `conda --help`
- `conda [command] --help`

`conda info`

- Display information about current conda install
- `conda info --all`
- `conda info --envs`
- `conda info --system`

Conda default “anaconda” environment

- `conda create -n <env> anaconda`
- numpy
- pandas
- matplotlib
- lots of stuff

`conda create` a new environment

- Create a new conda environment from a list of specified packages
- `conda create -n <env> python`
- Must supply at least one package
- Lots of optional arguments

conda environments continued...

- More realistic example
- `conda create -n <env> python=2 numpy matplotlib ipython ipython-notebook netcdf4`

Activating environment

- Unix : `source activate <env>`
- Windows: `activate <env>`

conda install into an environment

- Install a list of packages into a specified conda environment
- `conda install -n <env> matplotlib`
- Dealing with specific package versions
- `conda install -n <env> matplotlib=1.2`

conda list

- List packages in a conda environment
- `conda list`

Sharing & reproducing science w/ `conda list --export`

- `conda list --export > exported_packages.txt`
- share your exported_packages.txt w/ colleague
- `conda create -n <env> --file exported_packages.txt`

conda update

- Update conda packages
- `conda update --all` to update all installed packages in the environment
- Conda can self-update

```
conda update conda
```

```
conda update anaconda
```

`conda config`

- Modify configuration values in .condarc
- `conda config --add channels rsignell`
- `conda config --get channels --system`

source deactivate

- To deactivate the environment

Removing an environment

- `conda remove --all -n <env>`

Binstar

- `https://binstar.org`
- Package hosting server that works w/ conda
- Often, consuming packages via binstar
- Can also distribute packages via binstar

Binstar channels

- Channels are tied to **users** or **organizations**
- `https://binstar.org/unidata`
- `https://binstar.org/risgnell`
- Channels can be added to conda configuration (.condarc) so you can find packages of interest

`binstar` command utility

- Command line interface for `binstar`

Asking binstar for help

- `binstar --help`
- `binstar [command] --help`

`binstar search` & `binstar show`

- Search binstar for packages
- `binstar search proj4`
- `binstar show SciTools/proj4`

Sharing your work/APIs/packages via binstar

- Create an account at binstar.org
- Create recipe
- Create package
- Upload to binstar

Steps for uploading package to binstar in more detail

- Create recipe directory
- Create meta.yaml
- Create build.sh or bld.bat
- `conda build` package
- Upload to binstar

Example recipes

- Best is to follow examples at `https://github.com/conda/conda-recipes`

directory layout for conda recipe

```
`-- siphon
    |-- bld.bat
    |-- build.sh
    |-- meta.yaml
```

build.sh and bld.bat

- Typically a very simple file
- Contains build instructions

example `build.sh` and `bld.bat`

- bld.bat `"%PYTHON%" setup.py install`
- build.sh `$PYTHON setup.py install`
- For something written in C could be a bit more complicated invoking `make`

meta.yaml in more detail

- Human readable data format similar to XML
- Metadata that simply describes the build recipe
- Follow examples at https://github.com/conda/conda-recipes

example meta.yaml

```
package:
  name: siphon
  version: 0.3
source:
  git_url: https://github.com/Unidata/siphon
  git_tag: 0.1
build:
  number: 0
requirements:
  build:
    - python
    - setuptools
  run:
    - python
about:
  home: https://github.com/Unidata/siphon
  license: MIT
  summary: 'A collection of Python utilities for interactin
```

conda build

- Build from the parent of the recipe directory
- `conda build <package>`
- If successful, will give instructions on how to upload to binstar

binstar upload & share

- `binstar login`
- `binstar upload <package>`
- Tell colleagues about your channel so that they may use your work