Ana/Mini/Conda

or “How to completely wipe and reinstall your complete scientific Python stack in under 5 minutes” (if you have to)

Material at

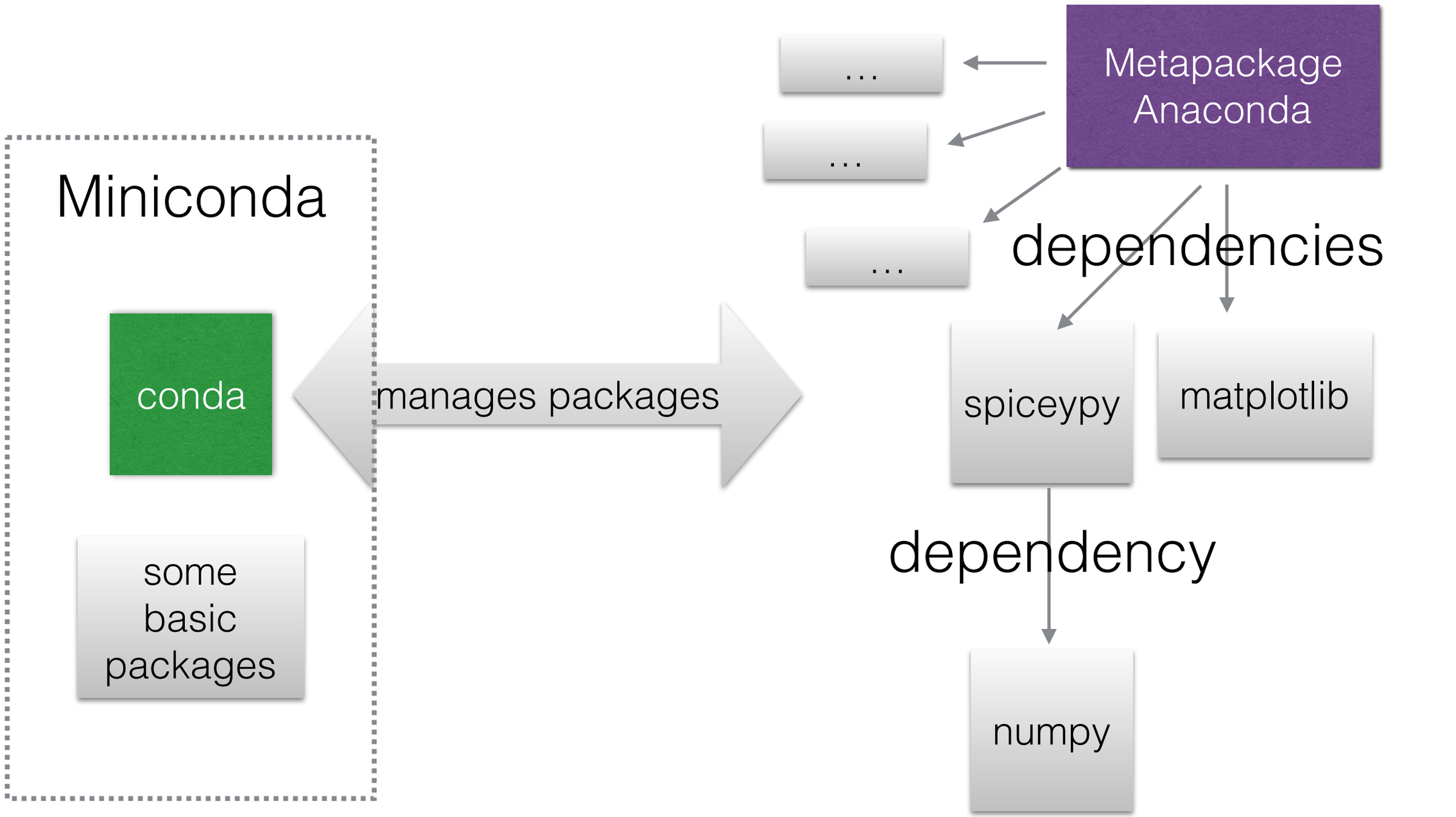
https://github.com/michaelaye/planetdata3

Conda vs Anaconda vs Miniconda

- How many here uses any conda-based system?

- How many really understand the difference between the 3 xxx-condas?

Conda vs Anaconda vs Miniconda (2)

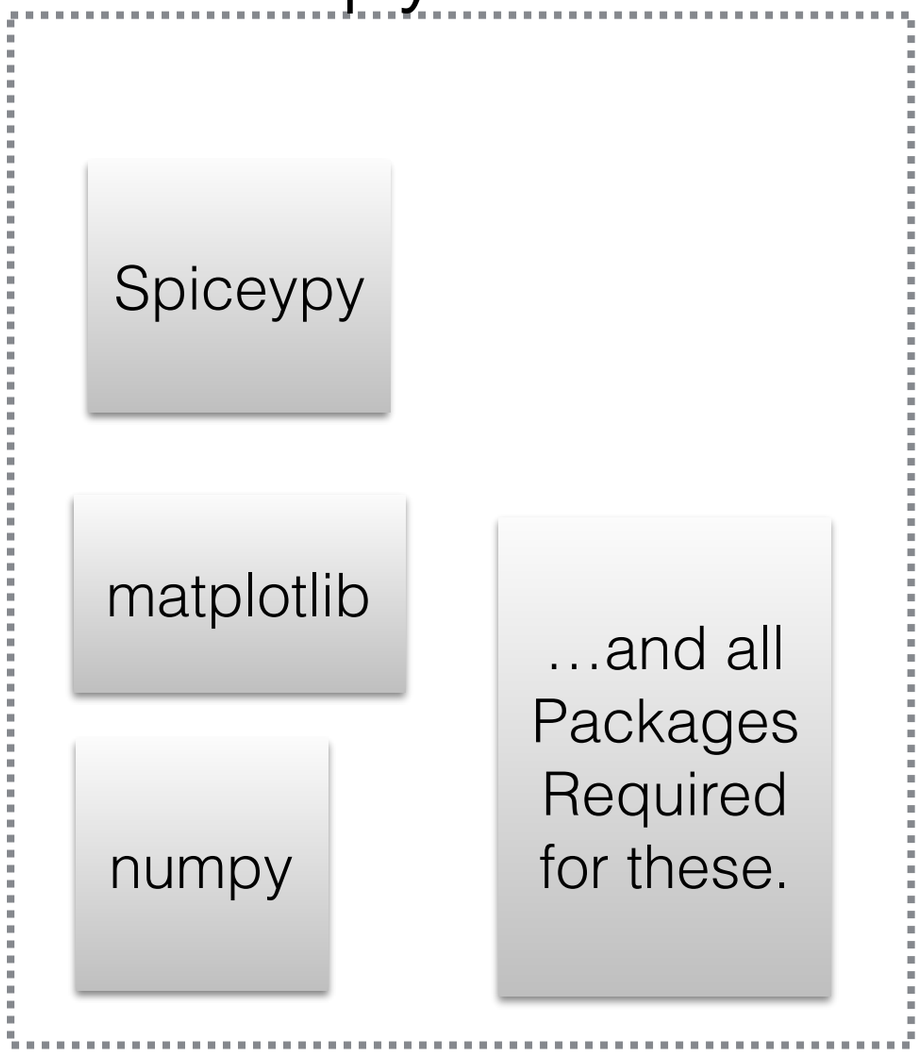


Standard conda package (grey)

Conda also manages virtual envs

If you only ever need one environment, you can stay with graphical Conda installer

conda create -n py37 python=3.7 spiceypy matplotlib numpy



conda create -n py2 python=2 some_weird_package_only_ running_under_python2

Use “conda activate <env-name>” to switch between environments.

Important: The conda package **“nb_conda_kernels”** enables switching inside the Jupyter notebook!

Conda vs Anaconda vs Miniconda (3)

- So, in summary:
  - conda is the executable that manages packages (not only Python, e.g. HDF binaries, FORTRAN libraries etc.)
  - “miniconda” is a minimum set of packages for proper operation of conda, installed into a “base”. **Use this if you understand conda well**.
  - “anaconda” is a meta-package with a huge list of scientific packages (dependencies) (**Recommended for beginners**)
  - Hence: after installing miniconda and executing “conda install anaconda”, you have the same python env as somebody that DL-ed the Anaconda distribution.

conda environments

- Advise: Don’t use the conda base environment for general work.
    - Eventually some of your installs (or Anaconda, Inc.) will mess up something.
- Always create a new default environment:
    - conda create -n ‘py37’ python=3.7
    - conda activate py37
- Find packages:
    - conda seach <package_name>
    - If list shows what you need:
    - conda install <package_name> (will also drag in dependencies)

conda environments

- What if a Python package is not available conda?
  - pip install pkg_name
    - **NOTE: Always do conda activate <env_name> before this (or anything really).** Because otherwise a different "pip" command might be used on your computer and install goes somewhere else.
- What if pkg not even on Pypi server?
  - Github clone && python setup.py install (or "develop")
- I use this mix for years with success

How to reinstall env in 5 min

- Even a “stable” env is rotting at some point
- The trick is to have:
  - File with a list of your conda packages
  - File with a list of your pip packages
  - File with a list of your own package folders and GH installs
  - a (couple of) bash script(s)

How to reinstall env in 5 min(2)

```
#!/bin/sh
if [ "$#" -ne 1 ]; then
    echo "Usage: $0 conda_env_name" >&2
    exit 1
fi
if [ -n $CONDA_DEFAULT_ENV ]
then
  if [ "$CONDA_DEFAULT_ENV" == "$1" ]
  then
    echo "Deactive $1 environment first." >&2
    exit 1
  fi
fi

conda env remove -n "$1" -y
conda create -n "$1" -y python=3
conda install -n "$1" -y --file ~/Dropbox/standard_py3_conda_packages.txt
cd ~/Dropbox/src/
source activate "$1"
```

- source deactivate
- ./reinstall_env.sh stable
- source activate stable
- ./install_my_libs.sh

```
1  #!/bin/sh
2  # reinstall all my packages
3  cd ~/Dropbox/src
4  for folder in 'pyciss' 'planet4' 'pyuvis' 'hirise_tools' 'nbtools' 'planetpy' 'pysis' 'p4terrains';
5      do cd $folder;
6      echo "Installing $folder";
7      echo;
8      pip install -e .;
9      cd .. ;
10     echo;
11 done
12 # reinstall packages from pip that are needed
13 pip install -r pip_packages_to_install.txt
14
```

conda channels

- Basically, channels are different repositories of packages
- Sometimes you see “conda install -c conda-forge pkg_name (or any other channel name)
  - I would not do that in “stable”, better to create a new env for testing first.
  - Mixing packages from channels often works, but sometimes not
    - e.g. GDAL will break when mixing “defaults” and “conda-forge” channel !!