Chris Calloway for

Triangle Python Users Group

at Caktus Group

December 14, 2017

What Is Conda

- Cross-platform
- Language Agnostic
- Package Manager
- Dependency Manager
- Environment Manager
- Package Creator
- Command Line Interface to the Conda Community

What Is Conda

- Cross-platform:
  - Linux
  - OSX
  - Windows

What Is Conda

- Language Agnostic:
  - Python
  - R
  - Ruby
  - Lua
  - Scala
  - Java
  - JavaScript
  - C/ C++
  - FORTRAN

What Is Conda

- Package Manager:
  - Replaces pip
  - Pip still works with Conda

What Is Conda

- Dependency Manager:
  - Replaces setuptools
  - Setuptools is still useful for making Conda-installable packages

What Is Conda

- Environment Manager:
  - Replaces virtualenv and venv
  - Virtualenv and venv don’t work with Conda

What Is Conda

- Package Creator:
  - Conda recipes create Conda packages
  - Conda recipes can wrap a setup.py

What Is Conda

- Command Line Interface to the Conda community
  - https://anaconda.org/
    - A package repository
    - A social network
    - A directory of package “channels”
    - An Anaconda channel
    - A community conda-forge channel
    - Your channel
    - Your organization's channel

What Is Conda

- Open source
- Created by Continuum Analytics
  - Creators of the Anaconda Python distribution

What Is Conda

- Huge uptake in the scientific and data analytics communities
  - By virtue of inclusion in the Anaconda Python distribution
  - SciPy community
  - PyData community

What Is Conda

- Less uptake in the web community although:
  - Django
  - Flask
  - CherryPy
  - Pyramid
  - Etc.

Are all available through Conda

Installing Conda

- Conda comes with Anaconda
  - Download at https://www.anaconda.com/download/
  - Choose platform:
    - Linux
    - OSX
    - Windows
  - Choose architecture:
    - 64 bit
    - 32 bit
  - Choose Python version:
    - 3.6
    - 2.7

Installing Conda

- Chose installer type:
  - Graphical
  - Command Line
- Choose:
  - Full Anaconda (~500 packages in addition to the Python Standard Library)
  - Miniconda (Just enough to get started)
    - https://conda.io/miniconda.html
- Prepend to PATH
  - Optional at conclusion of installer
  - `export PATH="/Users/cbc/anaconda/bin:$PATH"`

Managing Environments

- Parent environment:

```
$ python
Python 3.6.0 |Anaconda 4.3.1 (x86_64)| (default, Dec
23 2016, 13:19:00)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-
600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for
more information.
>>>
```

Managing Environments

- Parent environment:

```
$ ipython
Python 3.6.0 |Anaconda 4.3.1 (x86_64)| (default, Dec 23
2016, 13:19:00)
Type "copyright", "credits" or "license" for more
information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's
features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for
extra details.

In [1]:
```

Managing Environments

- Parent environment:

```
$ jupyter notebook
[I 20:29:16.190 NotebookApp] Serving notebooks from local
directory: /Users/cbc
[I 20:29:16.190 NotebookApp] 0 active kernels[I
20:29:16.190 NotebookApp] The Jupyter Notebook is running
at:
http://localhost:8888/?token=7e10099bd7e3b1857b4b491b9509
db3d49789174ff456404
[I 20:29:16.190 NotebookApp] Use Control-C to stop this
server and shut down all kernels (twice to skip
confirmation).
…
[I 20:29:16.402 NotebookApp] Accepting one-time-token-
authenticated connection from ::1
```

CONDA

Managing Environments

- Parent environment:

```
$ which python
/Users/cbc/anaconda/bin/python
$
```

Managing Environments

- Create a child environment:

```
$ conda create -n example anaconda
```

- Generates a ton of output while also prompting for acceptance of package list
- The conda command takes subcommands

Managing Environments

- List environments:

```
$ conda env list
# conda environments
#
example       /Users/cbc/anaconda/envs/example
root       *  /Users/cbc/anaconda
$
```

- Conda subcommands may take their own subcommands
- The splat indicates the currently "active" environment

Managing Environments

- List subcommands:

  ```
  $ conda -h
  ```

  - info
  - help
  - list
  - search
  - create
  - install
  - update
  - upgrade
  - remove
  - uninstall
  - config
  - clean

- Includes a brief explanation of each subcommand

Managing Environments

- List subcommands of subcommands:

```
$ conda env -h
```

  - attach
  - create
  - export
  - list
  - remove
  - update
  - upload

- Shows both options and further subcommands and brief explanations of both

Managing Environments

- List further subcommands and options

```
$ conda env export -h
```

- Show both short and long form options

Activating Environments

```
$ which python
/Users/cbc/anaconda/bin/python
$ source activate example
(example) $ which python
/Users/cbc/anaconda/envs/example/bin/python
(example) $
```

- Notice the command prompt changes to include the name of the active child environment in parentheses

Activating Environments

- On Windows, simply:

```
$ activate example
(example) $
```

- `activate` is a batch file on Windows

Activating Environments

- All activation really does is munge your PATH environment variable so that the Conda environment is searched first:

```
(example) $ echo $PATH
/Users/cbc/anaconda/envs/example/bin:
/Users/cbc/anaconda/bin:
/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
(example) $ which python
/Users/cbc/anaconda/envs/example/bin/python
(example) $
```

Deactivating Environments

```
(example) $ source deactivate
$ echo $PATH
/Users/cbc/anaconda/bin:
/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
$ which python
/Users/cbc/anaconda/bin/python
$
```

- Notice the command prompt changes back to not include a child environment name
- Simply `deactivate` on Windows

Installing To Environments

- Installing packages is the primary reason you want to have a child environment
- Each environment has an **isolated** set of installed packages
- Multiple environments may have different versions of the same package without interfering with one another

Installing To Environments

- If the packages installed in an environment become problematic, you can simply remove the envionment
- It's a good idea not to install additional packages to your parent environment
- Environments can be recreated by exporting the list of installed packges to a file

Installing To Environments

- Let's create a new child environment and activate it:

```
$ conda create -n example1 python=2
$ source activate example1
(example1) $
```

- We've specified a particular version of package
- Installed packages can be binaries, not just Python packages
- Even a different Python binary than the parent Python
- Installing just Python and its dependencies is like installing miniconda

Installing To Environments

- Easy to manage multiple versions of Python

```
(example1) $ python
Python 2.7.13 |Continuum Analytics, Inc.| (default, Dec 20 2016, 23:05:08)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>>
```

CONDA

Installing To Environments

- Let's install Node.js in our child envrironment:

```
(example1) $ which node
(example1) $ conda install nodejs
... lots of output ...
(example1) $ which node
/Users/cbc/anaconda/envs/example1/bin/node
(example1) $ node
> .exit
(example1) $
```

- Conda is language agnostic

Installing To Environments

- Your package installations to your child envrironment are isolated to your child environment:

```
(example1) $ source deactivate
$ which node
$
```

- This is what makes Conda a superior alternative to other tools such as homebrew, ports, fink, yum, apt-get, and virtualenv

Uninstalling

- Let's uninstall Node.js from our child envrironment:

```
$ source activate example1
(example1) $ conda uninstall nodejs
 ... lots of output ...
(example1) $ which node
(example1) $
```

- Make sure you know which environment is activated!

Removing an Environment

- Or we could remove the entire child envrironment:

```
(example1) $ source deactivate
$ conda env remove -n example1
$ which python
/Users/cbc/anaconda/bin/python
$
```

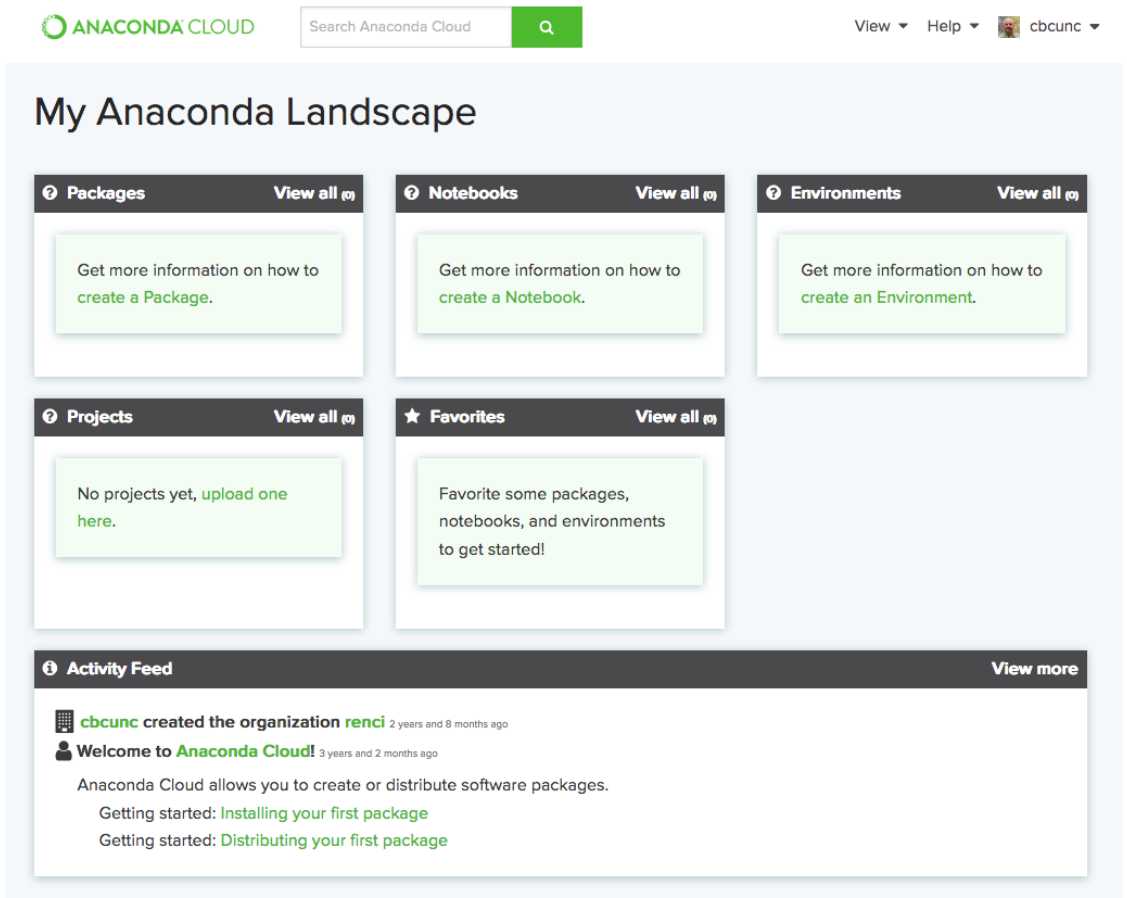
- Make sure the environment is deactivated!

Conda Channels

ANACONDA
CLOUD

CONDA

Conda Channels



CONDA

Conda Channels

ANACONDA CLOUD

Search Anaconda Cloud

View Help cbcunc

Profile

cbcunc

Contributor since Sep 18, 2014

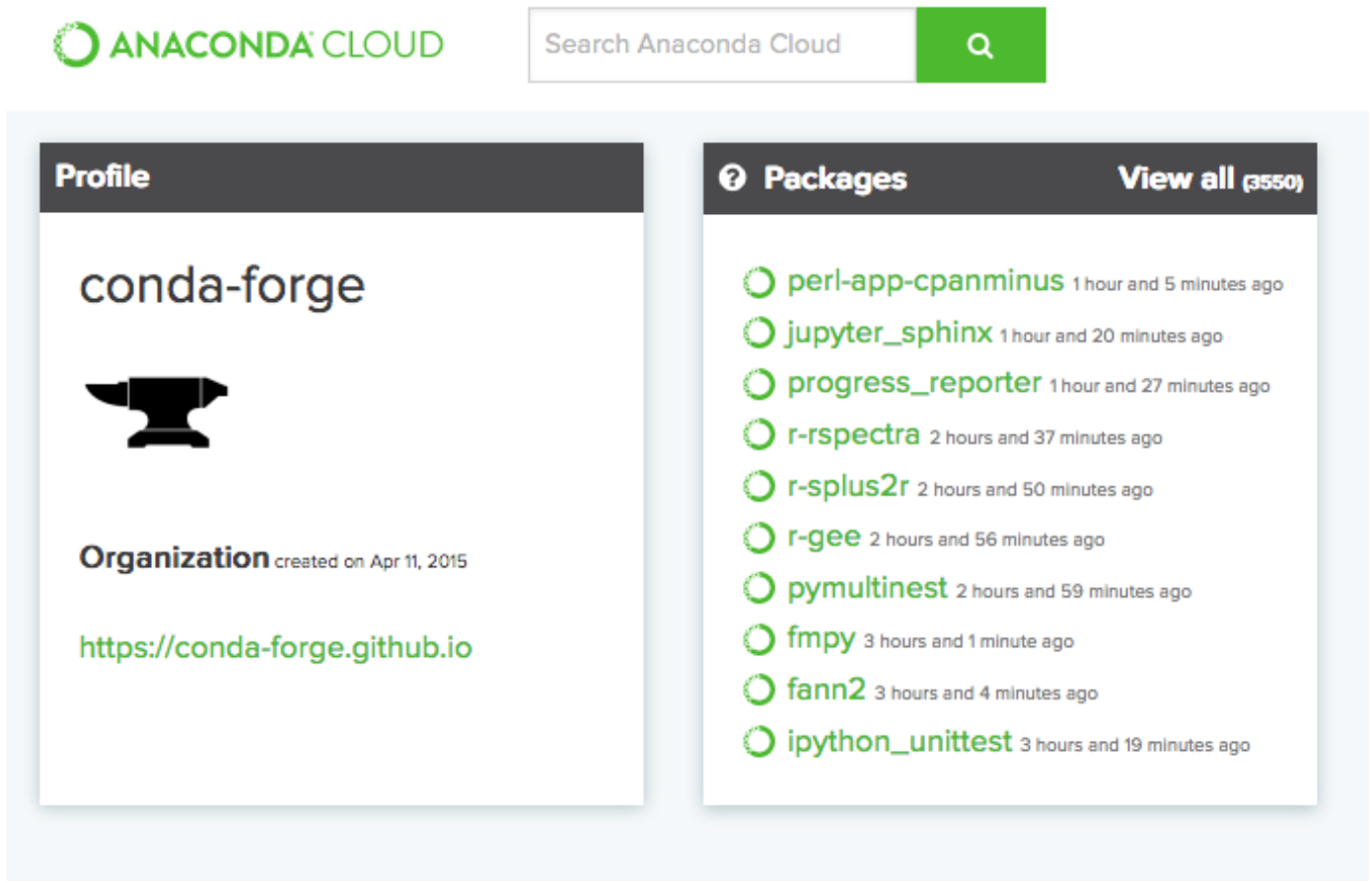
cbc@chriscalloway.org

Organizations

Looks like user **cbcunc** has nothing to share!

CONDA

Conda Channels



CONDA

Conda Channels

- Create a child environment entirely of packages from the conda-forge channel:

```
$ conda create -n example2 -c conda-forge pandas
Fetching package metadata ...........
Solving package specifications: .
Package plan for installation in environment
/Users/cbc/anaconda/envs/example2:
The following NEW packages will be INSTALLED:
    ca-certificates: 2017.11.5-0       conda-forge
    ......
    zlib:              1.2.11-0        conda-forge
Proceed ([y]/n)? Y
ca-certificate 100% |################| Time: 0:00:00   2.58 MB/s
......

zlib-1.2.11-0. 100% |################| Time: 0:00:00  19.44 MB/s
......
$
```

Conda Channels

- Dependencies will come from the default channels if not in the specified channel:

```
$ source activate example2
(example2) $ conda list
# packages in environment at /Users/cbc/anaconda/envs/example2:
#
ca-certificates   2017.11.5         0    conda-forge
certifi           2017.11.5   py36_0    conda-forge
......

numpy             1.13.1      py36_0
......
(example2) $
```

- Notice that numpy is coming from the default Anaconda channel, not conda-forge

Conda Channels

- The installed package list can be piped to a file:

```
(example2) $ conda list --explicit > example2.txt
(example2) $ cat example2.txt
(example2) $
```

- Similar to a requirements.txt file
- The –explicit option outputs URLs to the exact
  - Channel
  - Version

of each package

Conda Channels

- The package list file can be used to create a new environment:

```
(example2) $ conda create -n example3 --file
example2.txt
(example2) $ source activate example3
(example3) $ conda list
```

- You don't have to deactivate one environment before creating or even activating another
- This makes conda also a replacement for much of virtualenvwrapper
- The package list file can be committed to your version control system

Conda Channels

- You don't need a package list file to clone an environment:

```
(example3) $ conda create -n example4 --clone example3
(example3) $ source activate example4
(example4) $ conda list
```

Conda Channels

- You may search default channels by keyword:

```
(example4) $ conda search node
Fetching package metadata .........
node-webkit.      0.10.1      0      defaults
nodejs            4.2.6       0      defaults
                  6.10.3      0      defaults
(example4) $
```

- There are currently two supported node versions in the Anaconda default channel

Conda Channels

- You may add specific channels to your search:

```
(example4) $ conda search -c conda-forge node
Fetching package metadata .........
node-webkit       0.10.1       0        defaults
nodebook          0.2.1        py_0     conda-forge
nodeenv           1.2.0        py_0     conda-forge
nodejs            4.2.6        0        defaults
                  6.10.3       0        defaults
                  4.4.1        0        conda-forge
                  4.4.7        0        conda-forge
                  4.5.0        0        conda-forge
                  6.6.0        0        conda-forge
                  6.10.2       0        conda-forge
                  6.11.0       0        conda-forge
                  6.12.0       0        conda-forge
                  8.8.1        0        conda-forge
(example4) $
```

- Other channels may have many other versions of particular packages

Conda Channels

- You may add channels to be included in all commands:

```
(example4) $ conda config --add channels conda-forge
(example4) $ cat ~/.condarc
channels:
  - conda-forge
  - defaults
(example4) $
```

- Using conda config creates a .condrc hidden YAML file in your home directory
- Conda config is much like git config

Conda Channels

- You may see all your configuration:

```
(example4) $ conda config --show
... lots of output ...
(example4) $ conda config --describe
... lots of output ...
(example4)
```

- You may also see descriptions of all the available configuration keys

Conda Channels

- If you know a package name, you may show a lot of information about all the available versions of the package:

```
(example4) $ conda info iris
... lots of output ...
(example4) $
```

- Now that conda-forge was added to defaults in .condarc, we don't need to add –c channel to our conda commands to use the conda-forge channel
- Channels are searched in the order they appear in .condarc

Conda Channels

- If you know a package name, you may show a lot of information about all the available versions of the package:

```
(example4) $ conda info iris
... lots of output ...
(example4) $
```

- Iris is a meteorological package not in the Anaconda channel
- Many discipline specific packages are available in conda-forge
- Packages added to conda-forge are community reviewed and moderated

Conda Recipes

- Conda packages are built by a new conda command you install:

```
(example4) $ source deactivate
$ conda install conda-build
$ conda update conda
$ conda update conda-build
$
```

- The conda-build package adds several new conda subcommands

Conda Recipes

- Skeleton is one of those new subcommands:

```
$ mkdir Recipes
$ cd Recipes
$ conda skeleton pypi primer
$ ls primer
$
```

- Skeleton's purpose is to create boilerplate for your Conda recipe
- The recipe a directoy with at least a meta.yaml file
- The simplest way to create a pure Python recipe is from a distribution on the Python package index (pypi)
- Primer is an example Python distribution

Conda Recipes

- It is normal to edit the meta.yaml file created by skeleton:

```
$ emacs primer/meta.yaml
$
```

- Meta.yaml is metadata for your conda recipe
- The recipe directory may contain other files
- But only meta.yaml is required
- Meta.yaml keys are described in the online Conda docs
- Only a few are required

Conda Recipes

- The build subcommand creates a distribution from the recipe:

```
$ conda build primer
... lots of output ...
$ ls ~/anaconda/conda-bld
$
```

- Distributions are compressed file bundles
- Distributions are placed in your Anaconda parent's conda-bld subdirectory

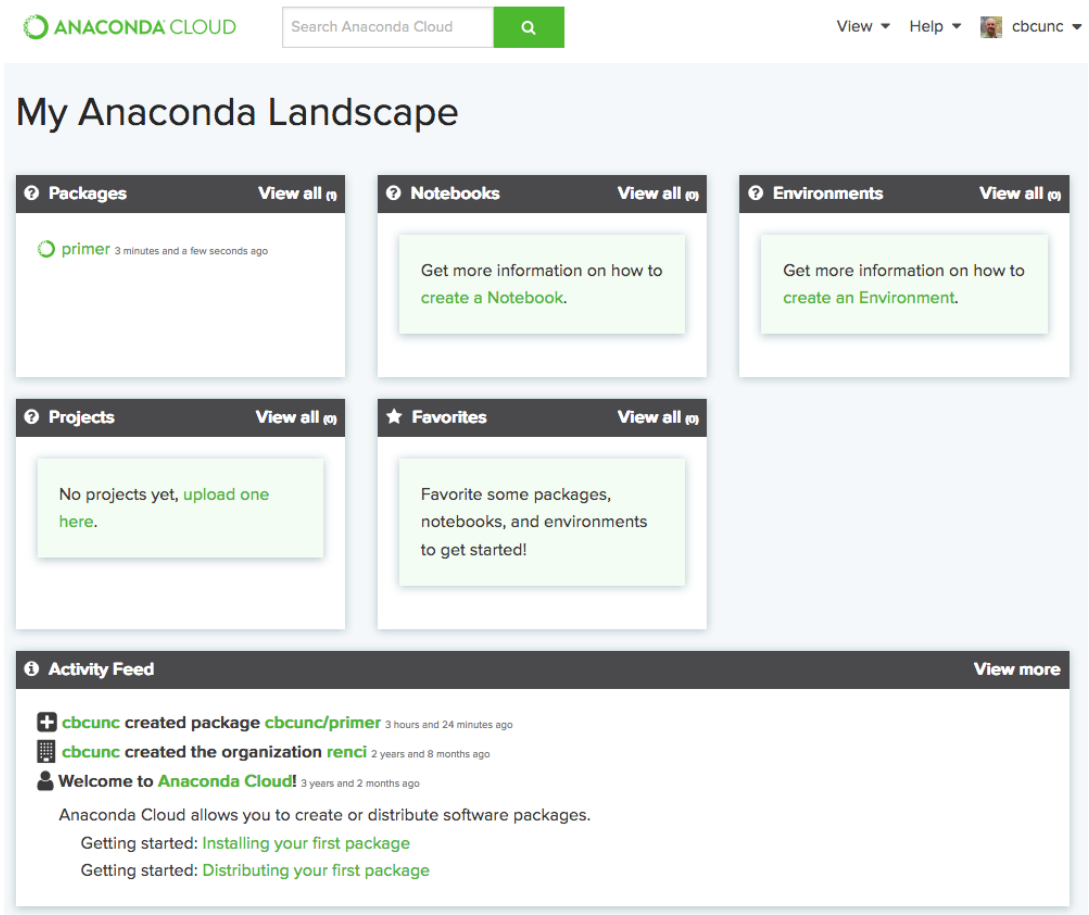
Conda Recipes

- The anaconda command manages Anaconda Cloud:

```
$ anaconda upload \
~/anaconda/conda-bld/osx-64/primer-1.4.4-py36_0.tar.bz2
... lots of output ...
$
```

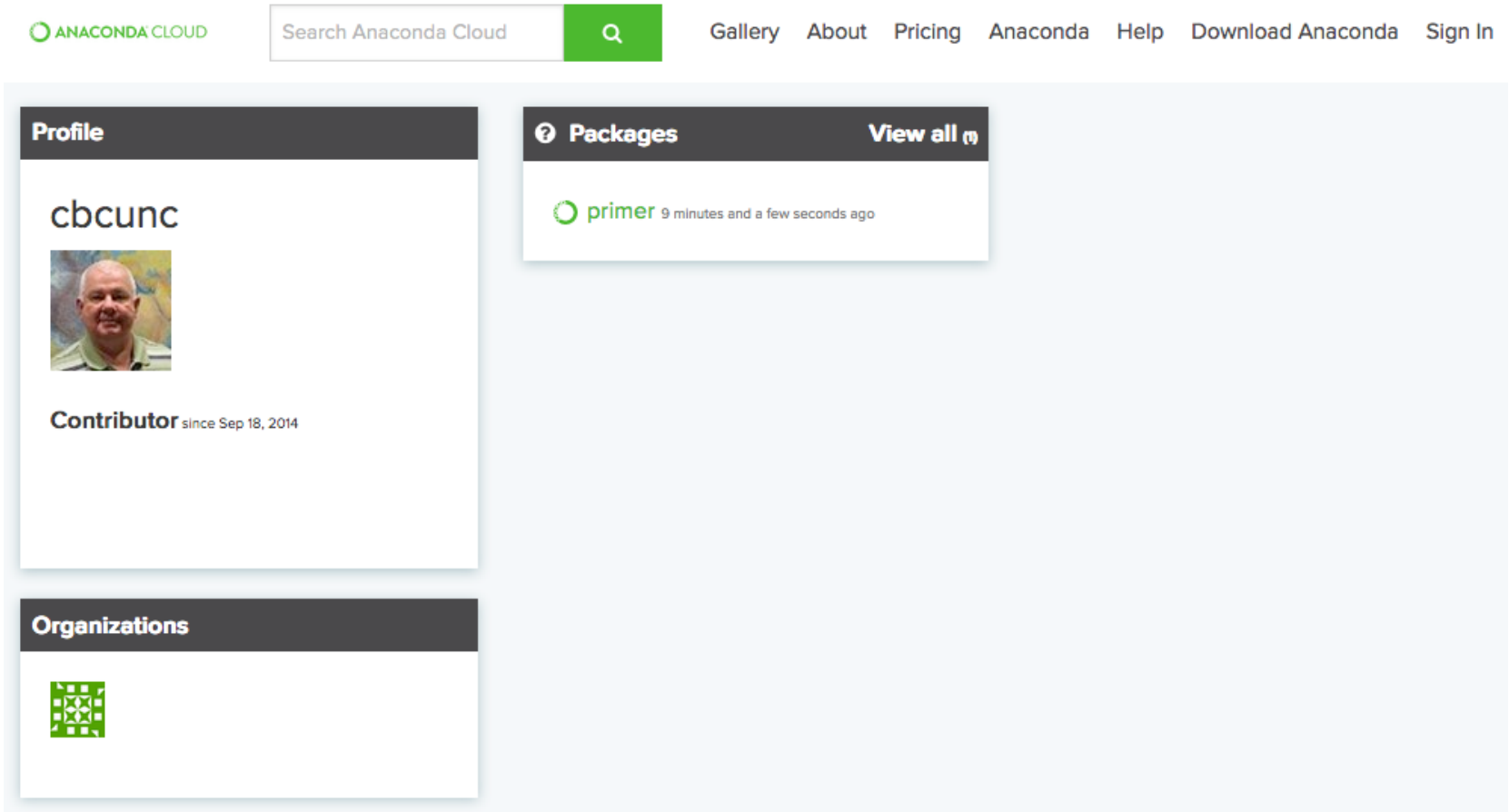
- Upload is one of the subcommands of the anaconda command
- You will need to authenticate with anaconda.org

Conda Recipes



CONDA

Conda Recipes



CONDA

Conda Recipes

- Let's create an enviroment in which to install the new distribution:

```
$ cd ~
$ conda create -n primer python
... lots of output ...
$ source activate primer
(primer) $ python
>>> import primer
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'primer'
>>>
```

Conda Recipes

- And let's install our new distribution into the new environment:

```
>>> exit()
(primer) $ conda install -c cbcunc primer
... lots of output ...
(primer) $ conda list
...
primer       1.4.4      py36_0      cbcunc
...
$
```

Conda Recipes

- The primer distribution is coming from a personal channel (cbcunc)
- You can install from public packages in anyone's channel
- Package may also be private or require authentication

Conda Recipes

- Awesome sauce:

```
(primer) $ python
>>> import primer
>>> primer.primes(10)
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
>>>
```

- Get more information at https://conda.io/