virtualenv(wrapper)

isolated environments for Python

Dave Jeffrey @jeffrhodeisland

The problem...

There are many ways to install third-party code for Python

- `easy_install` / `pip` and PyPI
- OS package managers, e.g. `sudo apt-get install django`
- Manual installs
- .egg files
- Just drop it into your application

easy_install / pip

`pip / easy_install` are similar to `gem` for Ruby or `npm` for Node.js. They're similarly very easy to use:

```
$ pip install django
$ python

>>> import django
```

Drawbacks

- Installs globally (may require root)
- Can't run different versions of the same library
- Pollutes the global environment

Virtualenv

Using `virtualenv`, we can create an isolated environment. We can install it with `pip`:

```
$ pip install virtualenv
```

Using virtualenv

We create a new environment as follows:

```
$ virtualenv env (--no-site-packages)
```

Activate environment

We need to 'activate' the environment before we can use it

```
$ source env/bin/activate
```

Using the new environment

Anything we install with pip now will only install into our environment, not globally

```
$(env) pip install django
```

Deactivating the environment

We can easily deactivate the new virtual environment

```
$(env) deactivate
```

And removing it and anything installed to it is as easy as

```
$ rm -r env
```

Some example use cases

- Virtual environments work well with pip's 'freeze' feature.
- Great for deploying websites (e.g. no need to upgrade every Django install on a shared server to use the latest version on one site)
- Good for experimenting with new packages

However...

It can be a little unwieldy:

- Each environment lives in its own folder
- Can clutter up projects / source control

Can we make it even easier?

Virtualenvwrapper

From the virtualenvwrapper site:

- Organizes all of your virtual environments in one place.
- Wrappers for creating, copying and deleting environments, including user-configurable hooks.
- Use a single command to switch between environments.
- Tab completion for commands that take a virtual environment as argument.

Simple workflow

```
$ mkvirtualenv mynewenv

$ workon mynewenv

$(mynewenv) pip install ...

$(mynewenv) lssitepackages

$(mynewenv) deactivate

$ rmvirtualenv mynewenv
```

And lots of other useful things

See more at:

http://www.virtualenv.org

http://www.doughellmann.com/projects/virtualenvwrapper