An Introduction to the Virtualenv Sandbox

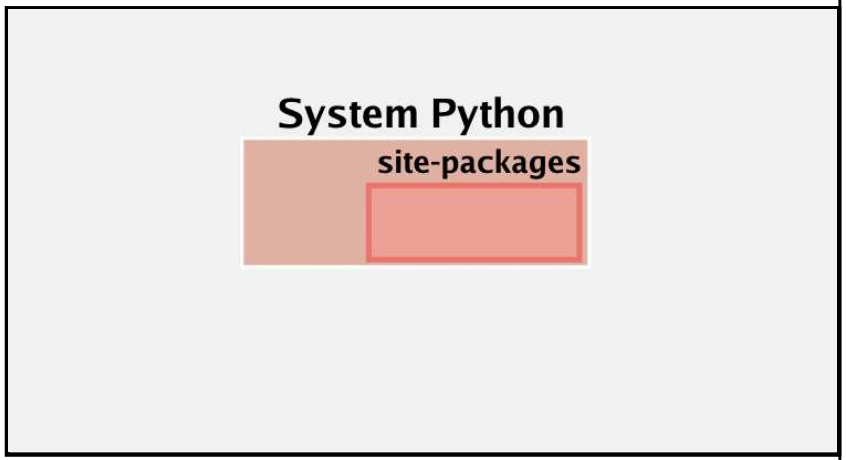
Author: Jeff Rush <jeff@taupro.com>

Date: March 25, 2009
Series: Python Eggs and Buildout Deployment - PyCon 2009 Chicago

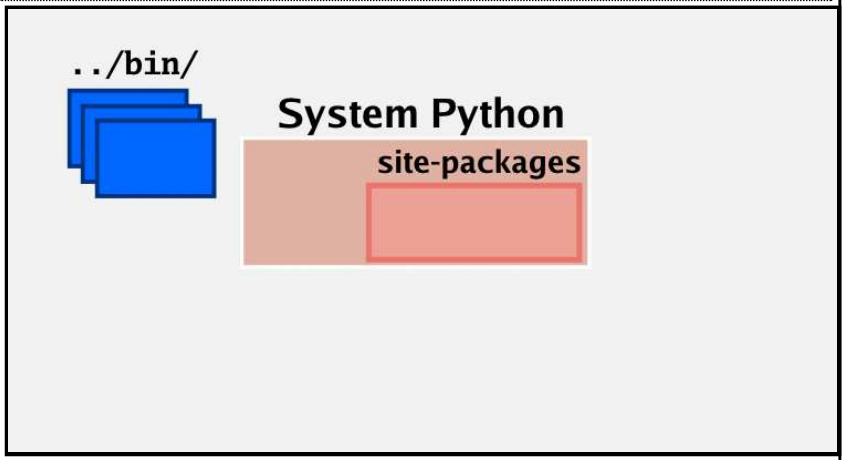
A brief getting-started lesson in using the *virtualenv* tool to sandbox your Python development work and, as a prerequisite, the steps for setting up EasyInstall to download it. Use of a sandbox to insulate each project from the others makes them easier to manage and experiment with.

Topic Roadmap

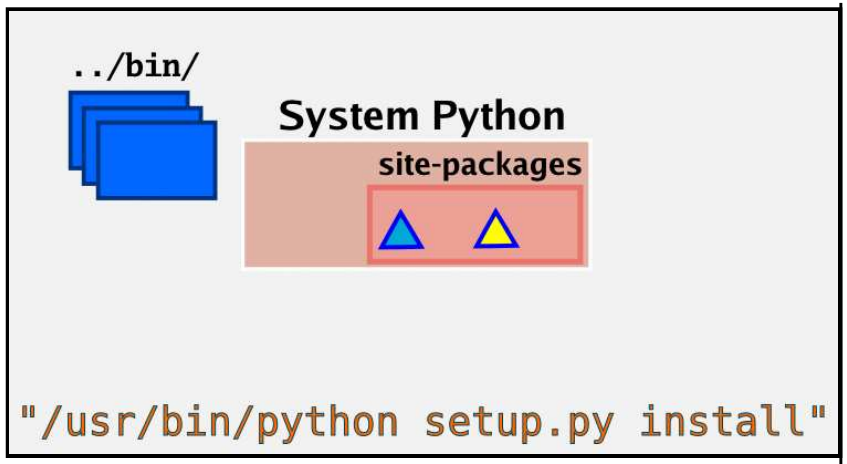
Conceptual Overview

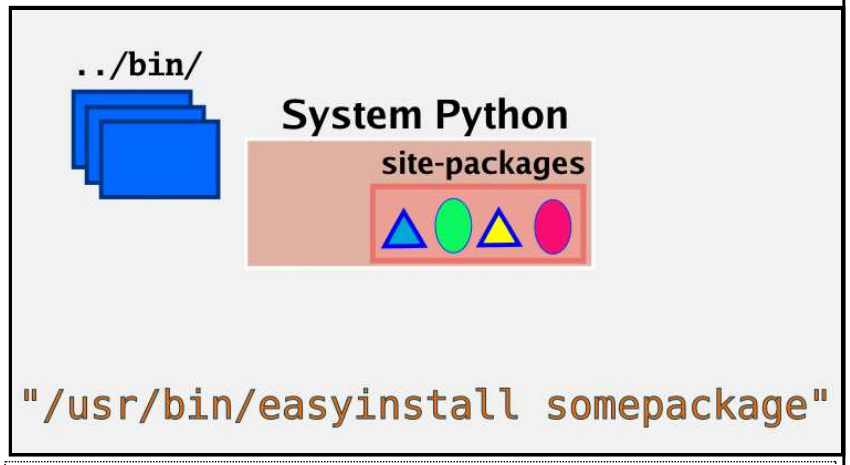


In a conventional environment, we have just the "system Python", with its global site-packages, which also requires Administrator privileges to install/deinstall.

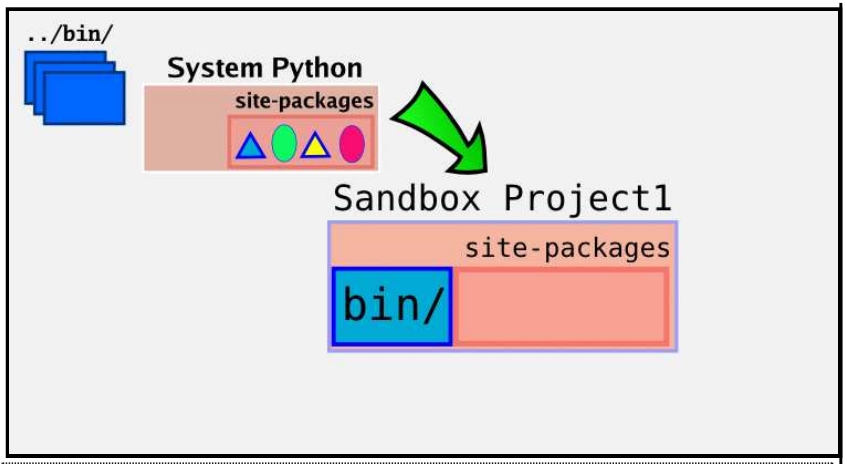


Any scripts installed by packages reside in one or more system-wide directories, providing an opportunity for collision if not named uniquely or taking versions into account.

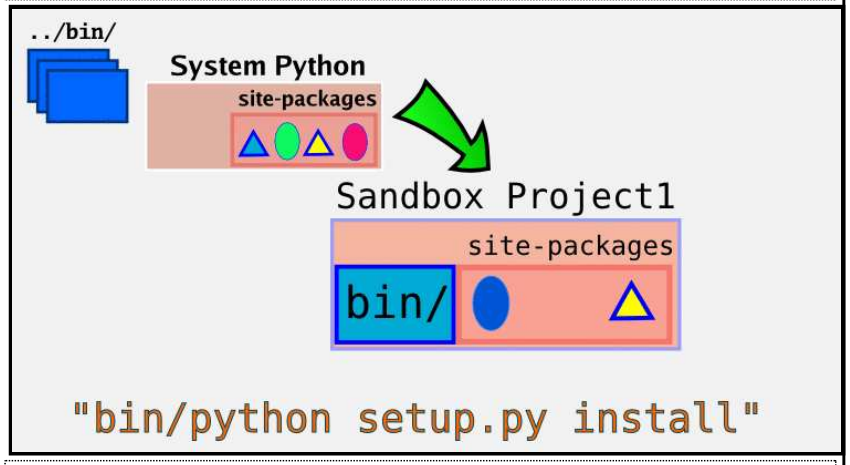




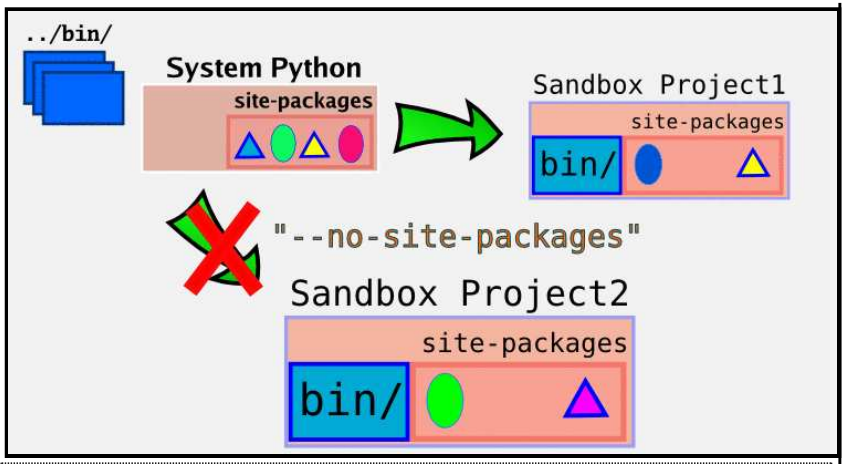
When you stall a package, they get all mixed together in the single *site-packages* directory, making it hard to coordinate with other developers or to use more than one version of a package at a time.



Virtualenv enables a different model of organizing your packages, by enabling separate installation areas or *sandboxes*.



Virtualenv supports packages based on both distutils (*setup.py install*) and setuptools (*easyinstall eggs*).



Notice that there is a link between the sandbox and the system Python. Normally your *virtualenv* sandbox is isolated from your system Python, but requests for modules not found within the sandbox flow through to the system Python. This means if you install extra software into the system Python it will automatically become available in all *virtualenv* sandboxes.

There is a command-line option, **--no-site-packages** that changes this behavior to exclude the system Python from all failing searches, for stronger isolation at the expense of having to explicitly install more dependencies.

Facts About Virtualenv

- written by Ian Bicking
- requires at least Python 2.3.5
- creates a fresh, isolated Python environment
- except it pre-installs *setuptools*
- must run programs from **bin/** (Scripts/)
- may be
  - in front of system Python
  - or totally replace system Python

Written by Ian Bicking, *virtualenv* allows you to set up isolated Python environments, each with their own site-packages directory. This makes it a good choice for experimenting with new packages or to deploy different programs with conflicting library requirements.

To maintain the isolation, Python programs must be run from the "bin/" subdirectory ("Scripts/" under Windows) within each sandbox.

To simplify the installation of *virtualenv*, we'll first install *EasyInstall*, bundled with *setuptools*, which I'll cover much more in-depth in the section on eggs.

You need a relatively modern version of Python, and access to the Internet for retrieving the necessary files. Because we'll be installing this system-wide (i.e. the sandboxing tools cannot themselves be inside the sandbox), you also need administrator privileges on your system.

These two packages, *EasyInstall* and *Virtualenv*, are being installed into the system site-packages directory, of the instance of Python with which you invoke it. This is true of a lot of tools we'll use today.

For those with multiple versions of Python on their system, to distinguish beween them, the tools are installed with a suffix of the version of Python used.

Installing Virtualenv from an Egg

Installation Steps:

```
$ cd /tmp
$ wget http://peak.telecommunity.com/dist/ez_setup.py
$ sudo python ez_setup.py
$ sudo easy_install virtualenv
```

Using your favorite tool, download **ez_setup.py** into a temporary directory and run it. This will download and install the appropriate setuptools egg for your Python version, and create a new system command *easy_install*.

Virtualenv can also be installed without using the egg by downloading, unpacking and running the *virtualenv.py* command specifically. However by installing the egg into the system Python directory, it makes the virtualenv command available system-wide.

Note: Windows users; do NOT put **ez_setup.py** inside your Python installation. Use a temporary directory elsewhere.

The "sudo" command is the Unix-way of running the "easy_install" command with administrator privileges.

Creating and Using Sandboxes

```
$ virtualenv pycon2009
$ virtualenv --no-site-packages pycon2009
$ cd pycon2009
$ bin/python

$ source bin/activate
$ deactivate
```

"virtualenvwrapper" Article by Doug Hellmann

To create a sandbox, run the *virtualenv* command and pass to it the pathname of a new directory in which you want it to reside.

Since you created the directory yourself, you have all the permissions you need to install new modules or libraries into the environment. And since the environments are light weight and easy to create, you can have as many of them as you want.

If you want stronger isolation from the system Python, use the **--no-site-packages** option to omit the system packages from the search path for your sandbox.

To run within the sandbox, run the Python interpreter in the bin directory. If you prefer to avoid typing the "bin/" prefix, *virtualenv* provides the "activate" command to rewrite the search paths for your shell session. The command "deactivate" reverses these changes.

Warning: the "activate" command for virtualenv has nothing to do with the concept of activating or de-activating a Python egg, which means placing it onto the sys.path using a .pth file.

Warning: On Windows you shouldn't create a *virtualenv* sandbox in a path with a space in any name.

Virtualenv has a few hooks which can be used by a custom bootstrap script. This can be useful to automatically install a few additional packages after creating a virtual environment.

Directory Layout of a Sandbox

```
bin/
  activate
  easy_install
  easy_install-2.5
  python

lib/
  python2.5/
    site-packages/
    distutils/
      distutils.cfg
```

Note: On Windows, the directory names are slightly different, with "Scripts/" being used for "bin/", and "Lib/" being used for "lib/".

Warning: Virtualenv is currently incompatible with a system-wide **distutils.cfg** and per-user **~/.pydistutils.cfg**. If you have either of these files, *virtualenv* will put the **easy_install** command into the **bin/** directory specified in that config file, rather than into the sandbox where it belongs.

Limitations of Virtualenv

- easy to experiment, hard to repeat and manage
- no simple way to list what you've installed
- no simple way of uninstalling packages
- the *buildout* tool is opposite to this
- and has the ability to manage non-Python aspects
- bakes in absolute filesystem paths

Further Reading

- Place to Download EasyInstall bootstrapper

  wget http://peak.telecommunity.com/dist/ez_setup.py

- Home Page for VirtualEnv

  http://pypi.python.org/pypi/virtualenv

- Article: "virtualenvwrapper" by Doug Hellmann

  http://tinyurl.com/6cfv82

- the Distutils-SIG and Mailing List

  http://www.python.org/sigs/distutils-sig/
  http://mail.python.org/pipermail/distutils-sig/

- Questions?

For those who prefer not to use tinyurl.com, the complete URL for Doug Hellmann's article is:

http://www.doughellmann.com/articles/CompletelyDifferent-2008-05-virtualenvwrapper/index.html