Entornos virtuales

WTPC 2019

Virtual environments: ¿para qué?

Crear y administrar distintos entornos.

Tenemos dos proyectos: A y B

- A depende de la librería C, versión 1.0
- B depende de la librería C, versión 2.0

Python no tiene forma de diferenciar las dos versiones, y va a utilizar la misma versión para los dos proyectos.

¿Entonces? Utilizamos un entorno virtual para cada proyecto.

Virtual environments: pipenv

Combina Pipfile, pip y virtualenv en un único comando

Instalación

```
$ pip3 install pipenv
```

pipenv: ¿cómo utilizarlo?

```
$ cd proyecto
$ pipenv shell
Creating a virtualenv for this project…
Pipfile: /home/rodrigo/proyecto/Pipfile
Using /usr/bin/python3 (3.7.3rc1) to create virtualenv…
⠸ Creating virtual environment...Already using interpreter /usr/bin/python3
Using base prefix '/usr'
New python executable in ~/.local/share/virtualenvs/proyecto-D7nItVvg/bin/python3
Also creating executable in ~/.local/share/virtualenvs/proyecto-D7nItVvg/bin/python
Installing setuptools, pip, wheel...
done.

✔ Successfully created virtual environment!
Virtualenv location: /home/rodrigo/.local/share/virtualenvs/proyecto-D7nItVvg
Creating a Pipfile for this project…
Launching subshell in virtual environment…

$  . /home/rodrigo/.local/share/virtualenvs/proyecto-D7nItVvg/bin/activate
(proyecto) $
```

pipenv: ¿cómo utilizarlo?

```
(proyecto) $ ls
Pipfile

(proyecto) $ cat Pipfile
[[source]]                          Fuente de los paquetes
name = "pypi"
url = "https://pypi.org/simple"
verify_ssl = true

[dev-packages]                      Paquetes requeridos por los desarrolladores

[packages]                          Paquetes requeridos

[requires]                          Otros requerimientos
python_version = "3.7"
```

pipenv: instalar librerías

```
(proyecto) $ pipenv install numpy
(proyecto) $ pipenv install pytest --dev
(proyecto) $ pipenv install flask==0.12.1
```

```
(proyecto) $ pipenv uninstall numpy
```

pipenv: Pipfile y Pipfile.lock

```
(proyecto) $ ls
Pipfile  Pipfile.lock
```

pipenv: Pipfile

Se puede editar a mano

```
(proyecto) $ cat Pipfile
[[source]]
name = "pypi"
url = "https://pypi.org/simple"
verify_ssl = true

[dev-packages]
pytest = "*"

[packages]
numpy = "*"
flask = "==0.12.1"

[requires]
python_version = "3.7"
```

pipenv: Pipfile.lock

Permite crear entornos determinísticos: contiene la versión exacta de los paquetes, sus dependencias y hasta sus subdependencias.

NO DEBE editarse a mano

Se crea a partir del Pipfile:

```
(proyecto) $ pipenv lock
```

pipenv: dependencias

```
(proyecto) $ pipenv graph
Flask==0.12.1
  - click [required: >=2.0, installed: 7.0]
  - itsdangerous [required: >=0.21, installed: 1.1.0]
  - Jinja2 [required: >=2.4, installed: 2.10.1]
    - MarkupSafe [required: >=0.23, installed: 1.1.1]
  - Werkzeug [required: >=0.7, installed: 0.15.2]
numpy==1.16.3
pytest==4.4.1
  - atomicwrites [required: >=1.0, installed: 1.3.0]
  - attrs [required: >=17.4.0, installed: 19.1.0]
  - more-itertools [required: >=4.0.0, installed: 7.0.0]
  - pluggy [required: >=0.9, installed: 0.9.0]
  - py [required: >=1.5.0, installed: 1.8.0]
  - setuptools [required: Any, installed: 41.0.1]
  - six [required: >=1.10.0, installed: 1.12.0]
```

```
(proyecto) $ pipenv graph --reverse
```

pipenv: locación de los archivos

Locación del entorno virtual

```
(proyecto) $ pipenv --venv
~/.local/share/virtualenvs/proyecto-D7nItVvg
```

Locación del proyecto

```
(proyecto) $ pipenv --where
~/proyecto
```

¿Cómo guardar el entorno virtual en la carpeta del proyecto?

```
$ export PIPENV_VENV_IN_PROJECT=1
```

pipenv

¿Es necesario abrir el entorno virtual para correr un proyecto?

```
$ pipenv run <insert command here>
```