How to use a virtualenv in your web app (to get newer versions of django, flask etc)

A virtualenv (/pages/VirtualenvsExplained) is a way to create a python environment that's isolated and separate from the normal system-wide installed packages. It's particularly useful if you decide our 'default' versions of packages are not the versions you want to use -- to get the latest django, for example.

Using a virtualenv in your web app

You can use a virtualenv in a new web app (created using the “Manual configuration” option) or in your existing web apps. To use a virtualenv in your web app, do the following:

1. Create a virtualenv
2. Install packages into your virtualenv
3. Configure your app to use this virtualenv

Step 1: Create a virtualenv

Go to the **Consoles** tab and start a Bash console.

We recommend using *virtualenvwrapper*, a handy command-line tool, to create your virtualenv.

Specify which Python version to use for your virtualenv using the `--python` option, but note that it must match the version of Python you've chosen for your web app. So, to create a new Python 3.6 virtualenv, run this command:

```
$ mkvirtualenv myvirtualenv --python=/usr/bin/python3.6
```

...or similarly for Python 2.7:

```
$ mkvirtualenv myvirtualenv --python=/usr/bin/python2.7
```

You’ll see your virtualenv being created

```
Running virtualenv with interpreter /usr/bin/python3.6
Using base prefix '/usr'
New python executable in /home/myusername/.virtualenvs/myvirtualenv/bin/python3.6
Also creating executable in /home/myusername/.virtualenvs/myvirtualenv/bin/python
Installing setuptools, pip, wheel...done.
virtualenvwrapper.user_scripts creating /home/myusername/.virtualenvs/myvirtualenv/b
virtualenvwrapper.user_scripts creating /home/myusername/.virtualenvs/myvirtualenv/b
virtualenvwrapper.user_scripts creating /home/myusername/.virtualenvs/myvirtualenv/b
virtualenvwrapper.user_scripts creating /home/myusername/.virtualenvs/myvirtualenv/b
virtualenvwrapper.user_scripts creating /home/myusername/.virtualenvs/myvirtualenv/b

(myvirtualenv) $ which python
/home/myusername/.virtualenvs/myvirtualenv/bin/python
```

NOTE: If you see a `command not found` error when trying to run `mkvirtualenv` , you'll find some installation instructions here: InstallingVirtualenvWrapper (/pages/InstallingVirtualenvWrapper)

Once your virtualenv is ready and active, you’ll see `(myvirtualenv) $` in your prompt.

Step 2: Install packages into your virtualenv

Install the required packages into your virtualenv using `pip` . You can just use pip without the Python version number or `--user` flag.

```
$ workon myvirtualenv


(myvirtualenv) $ which pip   # this lets you check that the virtualenv has been activated
/home/myusername/.virtualenvs/myvirtualenv/bin/pip


(myvirtualenv) $ pip install django==1.7.1  # or flask, or whichever modules you want to use, optionally speci


Downloading/unpacking django==1.7.1
  Downloading Django-1.7.1-py2.py3-none-any.whl (7.4MB): 7.4MB downloaded
Installing collected packages: django
Successfully installed django
Cleaning up...
```

Step 3: Configure your app to use this virtualenv

Now that you have a virtualenv, and know its path, configure your app to use this virtualenv.

Go to the **Web** tab, and in the Virtualenv section, enter the path: `/home/myusername/.virtualenvs/myvirtualenv`

- TIP: if you're using virtualenvwrapper, you can just enter the name of the virtualenv, *myvirtualenv*, and the system will automatically guess the rest of the path (*/home/myusername/.virtualenvs etc*) after you hit ok.

Now, **Reload** your web app, and you should find it has access to all the packages in your virtualenv, instead of the system ones.

Deactivating and reactivating your virtualenv

Once you create your virtualenv, you need to activate it. It's automatically activated straight after you create it with `mkvirtualenv` , and you can re-activate it later with `workon` .

Re-activate using `workon` :

```
$ workon myvirtualenv
(myvirtualenv) $ which python
/home/myusername/.virtualenvs/myvirtualenv/bin/python
(myvirtualenv) $ python
Python 3.6.0 (default, Jan 13 2017, 00:00:00)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

To deactivate, use `deactivate` :

```
(myvirtualenv) $ deactivate
$ which python
/usr/bin/python
```

Want to improve this page? Submit a pull request! (https://github.com/pythonanywhere/help_pages)