Deploying an existing Django project on PythonAnywhere

Deploying a Django project on PythonAnywhere is a lot like running a Django project on your own PC. You'll use a virtualenv, just like you probably do on your own PC, you'll have a copy of your code on PythonAnywhere which you can edit and browse and commit to version control.

The main thing that's different is that, instead of using the Django dev server with `manage.py runserver` and viewing your site on localhost, you'll create what we call a *Web app* via the Web tab in our UI, and then configure it with a *WSGI file* whose job is simply to import your Django project.

And then your site will be live on the real, public Internet. woo!

Here's an overview of the steps involved.

1. Upload your code to PythonAnywhere
2. Set up a virtualenv and install Django and any other requirements
3. Set up your web app using the **manual config** option
4. Add any other setup (static files, environment variables etc)

Uploading your code to PythonAnywhere

Assuming your code is already on a code sharing site like GitHub or Bitbucket, you can just clone it from a **Bash Console**:

```
# for example
$ git clone https://github.com/myusername/myproject.git
```

That's the solution we recommend, but there are a few different methods documented on the uploading and downloading files (/pages/UploadingAndDownloadingFiles) help page.

Create a virtualenv and install Django and any other requirements

In your Bash console, create a virtualenv, naming it after your project, and choosing the version of Python you want to use:

```
$ mkvirtualenv --python=/usr/bin/python3.8 mysite-virtualenv
(mysite-virtualenv)$ pip install django
# or, if you have a requirements.txt:
(mysite-virtualenv)$ pip install -r requirements.txt
```

Warning: *Django may take a long time to install. PythonAnywhere has very fast internet, but the filesystem access can be slow, and Django creates a lot of small files during its installation. Thankfully you only have to do it once!*

TIP: *if you see an error saying* `mkvirtualenv: command not found` *, check out InstallingVirtualenvWrapper (/pages/InstallingVirtualenvWrapper).*

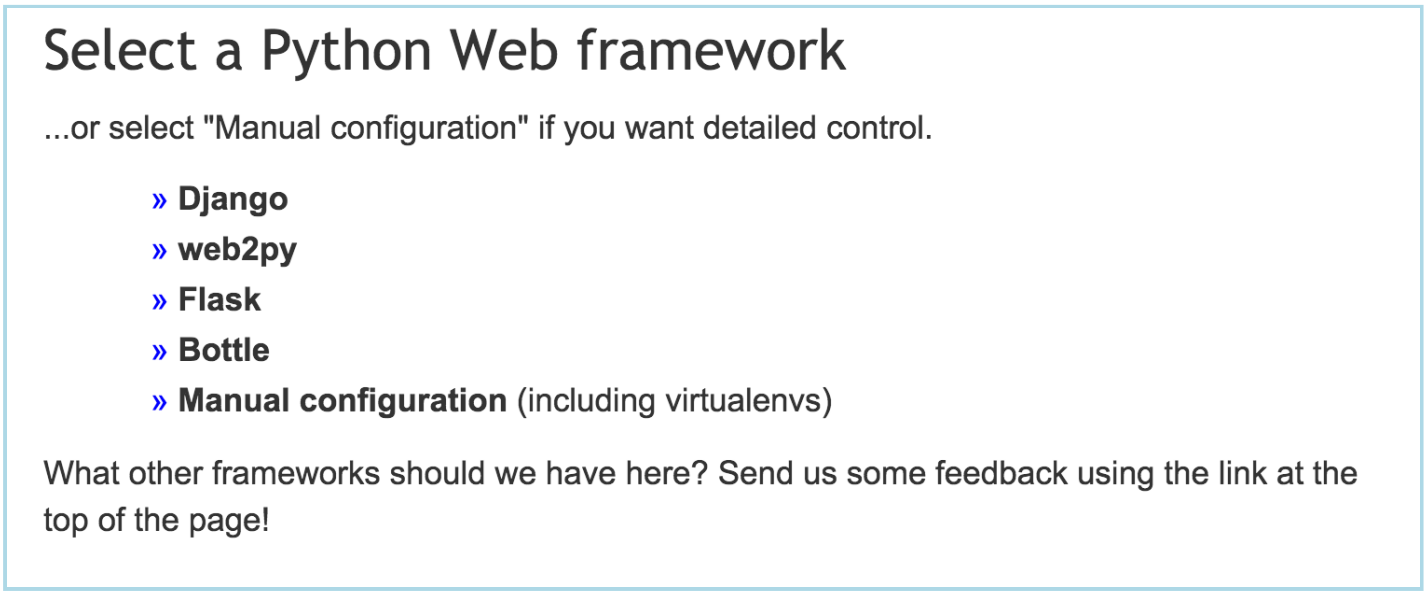
Setting up your Web app and WSGI file

At this point, you need to be armed with 3 pieces of information:

1. The path to your Django project's top folder -- the folder that contains "manage.py", eg */home/myusername/mysite*
2. The name of your project (that's the name of the folder that contains your settings.py), eg *mysite*
3. The name of your virtualenv, eg *mysite-virtualenv*

Create a Web app with Manual Config

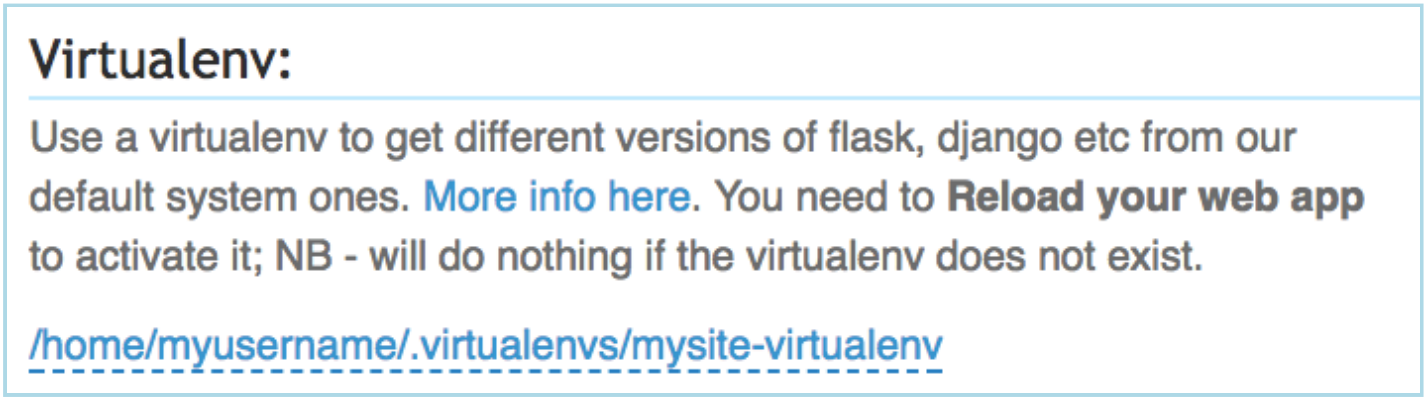
Head over to the **Web tab** and create a new web app, choosing the "Manual Configuration" option and the right version of Python (the same one you used to create your virtualenv).



- **NOTE:** Make sure you choose **Manual Configuration**, not the "Django" option, that's for new projects only.

Enter your virtualenv name

Once that's done, **enter the name of your virtualenv** in the Virtualenv section on the web tab and click OK.

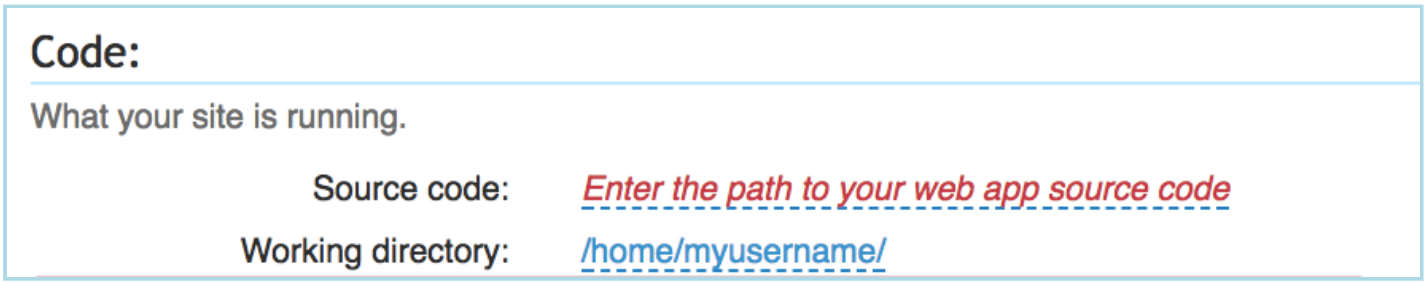


You can just use its short name "mysite-virtualenv", and it will automatically complete to its full path in /home/username/.virtualenvs.

Optional: enter path to your code

Although this isn't necessary for the app to work, you can optionally set your working directory and give yourself a convenient hyperlink to your source files from the web tab.

Enter the path to your project folder in the Code section on the web tab, eg */home/myusername/mysite* in **Source code** and **Working directory**



Edit your WSGI file

One thing that's important here: your Django project (if you're using a recent version of Django) will have a file inside it called `wsgi.py` . This is *not* the one you need to change to set things up on PythonAnywhere -- the system here ignores that file.

Instead, the WSGI file to change is the one that has a link inside the "Code" section of the **Web tab** -- it will have a name something like `/var/www/yourusername_pythonanywhere_com_wsgi.py` or `/var/www/www_yourdomain_com_wsgi.py` .

Click on the WSGI file link, and it will take you to an editor where you can change it.

Delete everything except the Django section and then uncomment that section. Your WSGI file should look something like this:

```
# ++++++++++ DJANGO ++++++++++
# To use your own Django app use code like this:
import os
import sys

# assuming your Django settings file is at '/home/myusername/mysite/mysite/settings.py'
path = '/home/myusername/mysite'
if path not in sys.path:
    sys.path.insert(0, path)

os.environ['DJANGO_SETTINGS_MODULE'] = 'mysite.settings'

## Uncomment the lines below depending on your Django version
###### then, for Django >=1.5:
from django.core.wsgi import get_wsgi_application
application = get_wsgi_application()
###### or, for older Django <=1.4
#import django.core.handlers.wsgi
#application = django.core.handlers.wsgi.WSGIHandler()
```

- Be sure to substitute the correct path to your project, the folder that contains manage.py, which you noted above.
- Don't forget to substitute in your own username too!
- Also make sure you put the correct value for `DJANGO_SETTINGS_MODULE`.
- This guide assumes you're using a recent version of Django, so leave the old `wsgi.WSGIHandler()` code commented out, or better still, delete it.

Save the file, then go and hit the **Reload** button for your domain. (You'll find one at the top right of the wsgi file editor, or you can go back to the main web tab)

Database setup

If, like most sites, your site uses a database, you'll need to set that up. Go to the **Consoles tab**, start a bash console, use `cd` to navigate to the directory where your Django project's `manage.py` lives, then run

```
./manage.py migrate
```

Checking it worked.

Go visit your site, it should be live! But it probably won't be using your CSS stylesheets, JavaScript and other things that are loaded from static files. To get those set up, check out the page configuring static files with Django (/pages/DjangoStaticFiles)

What to do if you see any errors.

First, check your **error log**. You'll find a link to it on your web tab.

Then check out the "debugging web app errors" section (/pages/#im-looking-at-an-error-message-in-my-web-app) from our help topics, particularly this article on sys.path and import errors (/pages/DebuggingImportError)

Additional configuration:

There are many different ways to set things up so that your database settings, `SECRET_KEY` , and so on are different in your local environment to the settings in your live environment on PythonAnywhere. If you're specifically using environment variables to store them, this page on setting environment variables for web apps (/pages/environment-variables-for-web-apps) will help.

Then, head over to the help page on Static files in Django (/pages/DjangoStaticFiles) for instructions on how to set up static file serving.

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