Chapter 20

Deploying a Project to Heroku

In this chapter you’ll style Learning Log, and then deploy it to Heroku’s servers. Once you’ve deployed the app, you’ll be able to access it from any device, and you’ll be able to share your work with others as well. This section will help you get your system set up for deploying projects to Heroku.

Making a Heroku Account

To make an account, go to <https://heroku.com/> and click one of the signup links. It’s free to make an account, and Heroku has a free tier that allows you to test your projects in live deployment.

Installing the Heroku Toolbelt

To deploy and manage a project on Heroku’s servers, you’ll need the tools available in the Heroku Toolbelt. To install the latest version, visit <https://toolbelt.heroku.com/> and follow the directions for your operating system. This will involve either a one-line terminal command or an installer you can download and run.

Installing required packages

You’ll also need to install a number of packages that help serve Django projects on a live server. In an active virtual environment, issue these commands:

(ll\_env)learning\_log$ pip install dj-database-url

(ll\_env)learning\_log$ pip install dj-static

(ll\_env)learning\_log$ pip install static3

(ll\_env)learning\_log$ pip install gunicorn

Issue the commands one at a time so you know if any package fails to install correctly. (Some of these packages may not install on Windows, so don’t be concerned if you get an error message when you try to install some of them. This shouldn’t affect your ability to deploy the project to Heroku.)

Updates

Heroku projects are stored in a directory called /app, and we use this fact to distinguish between local settings and Heroku-specific settings in the *settings.py* file. When you push a Django project to Heroku, the project is initially built from a temporary directory. If we only look for the /appdirectory, the project will fail to build properly.

There’s a simple fix for this. We’ll modify the if statement in *settings.py* to check for a /tmpdirectory as well as /app.

Checking for the /tmp directory

The if statement in *settings.py* looks like this in the book:

*# Heroku settings*

**if** os**.**getcwd() **==** '/app':

import dj\_database\_url

Here’s the updated version, which checks for /tmp directories as well:

*# Heroku settings*

cwd **=** os**.**getcwd()

**if** cwd **==** '/app' **or** cwd[:4] **==** '/tmp':

import dj\_database\_url

The actual /tmp directory is different for each deployment; it looks something like/tmp/build\_07bcdf3ae946245cce5541e622a3799a. This code looks at the first four characters of the current directory, and uses the Heroku settings if the current directory starts with /tmp.

With this update, your project should deploy without any issues. If you want to use WhiteNoise to manage your static files, read through the next section as well.

Using WhiteNoise to manage static files

The WhiteNoise project provides a better way to serve static files for Django projects. This project doesn’t significantly change the way Learning Log works, but if you create a project that sees a lot of traffic you’ll be better off with WhiteNoise.

If you haven’t worked through the second half of Chapter 20 yet, you can build your project with WhiteNoise. On page 466, under *Installing Required Packages*, install these packages instead:

(ll\_env)learning\_log$ pip install dj-database-url

(ll\_env)learning\_log$ pip install whitenoise

(ll\_env)learning\_log$ pip install gunicorn

After this your *requirements.txt* file will look slightly different. You’ll see whitenoise listed instead of dj-static and static3.

The *settings.py* file needs one additional line in the Heroku section. Add the STATICFILES\_STORAGEsetting, shown on the last line here:

*# Heroku settings*

cwd **=** os**.**getcwd()

**print**("--- CWD ---\n", cwd, "\n---\n")

**if** cwd **==** '/app' **or** cwd[:4] **==** '/tmp':

import dj\_database\_url

DATABASES **=** {

'default': dj\_database\_url**.**config(default**=**'postgres://localhost')

}

*# Honor the 'X-Forwarded-Proto' header for request.is\_secure().*

SECURE\_PROXY\_SSL\_HEADER **=** ('HTTP\_X\_FORWARDED\_PROTO', 'https')

*# Only allow heroku to host the project.*

ALLOWED\_HOSTS **=** ['\*']

DEBUG **=** True

*# Static asset configuration*

BASE\_DIR **=** os**.**path**.**dirname(os**.**path**.**abspath(\_\_file\_\_))

STATIC\_ROOT **=** 'staticfiles'

STATICFILES\_DIRS **=** (

os**.**path**.**join(BASE\_DIR, 'static'),

)

STATICFILES\_STORAGE **=** 'whitenoise.django.GzipManifestStaticFilesStorage'

Finally, the *wsgi.py* file needs to use WhiteNoise instead of Cling:

"""

WSGI config for learning\_log project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see

https://docs.djangoproject.com/en/1.10/howto/deployment/wsgi/

"""

import os

from django.core.wsgi import get\_wsgi\_application

from whitenoise.django import DjangoWhiteNoise

os**.**environ**.**setdefault("DJANGO\_SETTINGS\_MODULE", "learning\_log.settings")

application **=** get\_wsgi\_application()

application **=** DjangoWhiteNoise(application)

With these changes your project should run well on Heroku, even if you start to see higher levels of traffic.

If you’ve already built your project using dj-static and static3, you can uninstall these libraries and install whitenoise instead:

(ll\_env)learning\_log$ pip uninstall dj-static

(ll\_env)learning\_log$ pip uninstall static3

(ll\_env)learning\_log$ pip install whitenoise

Now you can reissue the pip freeze > requirements.txt command, and update *settings.py* and*wsgi.py*. Commit your changes using Git, and you’re ready to deploy your app to Heroku:

(ll\_env)learning\_log$ pip freeze > requirements.txt

(ll\_env)learning\_log$ git commit -am "Updated project to use WhiteNoise."

(ll\_env)learning\_log$ git push heroku master

Resources

If you want to know more about the deployment process, here are some helpful resources:

* [Getting Started on Heroku with Python](https://devcenter.heroku.com/articles/getting-started-with-python)
* [Django and Static Assets](https://devcenter.heroku.com/articles/django-assets)
* [Configuring Django Apps for Heroku](https://devcenter.heroku.com/articles/django-app-configuration)
* [The WhiteNoise project](http://whitenoise.evans.io/en/stable/index.html)

Questions

This is the most complicated project in the book, but it’s really satisfying to see your work deployed live on the internet. I really want you to be successful in your deployment! If you have an issue with the deployment process that you can’t seem to resolve, please get in touch:

Email: [ehmatthes@gmail.com](mailto:ehmatthes@gmail.com)

Twitter: [@ehmatthes](http://twitter.com/ehmatthes/)