Unit 3 / Lesson 3 / Project 4

Deploying Flask apps on Heroku

Estimated Time: 1-2 hours

In this assignment you'll learn how to host apps on Heroku by deploying your Hello World application from earlier in this lesson. Heroku is a platform-as-a-service (PaaS) that allows you to deploy web applications to virtual machines known as *dynos*. When someone visits your site, a dyno is accessed and used to serve your content.

First create a Heroku account if you don't already have one. To push your code to Heroku, you'll be using the Heroku Toolbelt, a command-line tool that interfaces with git and a few other things. If you're using Cloud9 you don't need to install anything because the Heroku Toolbelt is pre-installed.

After you have your Heroku account set up, you need to login to Heroku using the Heroku Toolbelt.

Login to Heroku by executing `heroku login` at the command line:

```
$ heroku login
Enter your Heroku credentials.
Email: frodo@example.com
Password:
Could not find an existing public key.
Would you like to generate one? [Yn]
Generating new SSH public key.
Uploading ssh public key /Users/frodo/.ssh/id_rsa.pub
```

Great! You're signed into Heroku at your command line. Now you're ready to create your first app. Make sure you're inside the *flask_hello_world* directory and then type in the terminal:

```
$ heroku create
Creating stark-fog-398... done, stack is cedar
http://stark-fog-398.herokuapp.com/ | git@heroku.com:stark-fog-398.git
Git remote heroku added
```

Heroku generates a random project name for you, and sets up a subdomain at http://example-app.herokuapp.com. If you go to the Heroku Dashboard you should see your app.

Now, before you can setup a dyno to host your app, you need a server that the dyno can use. There is a wonderful, lightweight, pure-Python server called Gunicorn that you can install with pip:

```
$ pip install gunicorn
```

To configure your dyno to properly use Gunicorn with your app, create a Procfile, short for "process file." The Procfile is simple:

```
$ echo 'web: gunicorn hello_world:app --log-file=-' > Procfile
```

This creates the Procfile and echos the instructions into it. You can open the Procfile or `cat Procfile` to verify. The `gunicorn` command takes the name of your module or package, in this case `hello_world`, followed by a colon, and then the name `app`, which will be imported from it. The `--log-file` switch just says "don't create a log file."

Finally, you need to specify your Python version. Check their list of supported Python runtimes and choose the latest supported Python 3 version; then declare that as your application's preferred runtime:

```
$ echo python-3.7.5 > runtime.txt
```

Be sure to replace `3.7.5` with the actual version number listed in Heroku's documentation.

Your project is now ready to push to Heroku, however you'll want to test it to make sure that it works. You can do this with the "heroku local" webserver, auto-installable by the Heroku toolbelt:

```
$ heroku local
```

The first time you run this, it will go out and download the necessary code; be patient. Once that's done, future usage will be a lot faster. This starts a webserver which you can view in your Cloud9 workspace with *Preview > Preview running application*

If everything works as it should, then you're ready to push your app to Heroku! First, kill the 'heroku local' process with Ctrl-c. If you haven't made a `requirements.txt`, you'll need one:

```
$ pip freeze > requirements.txt
```

It's best to save your progress in a git commit to master now. After you've done that, you can push your app to Heroku using git:

```
$ git push heroku master # Now you can push to Heroku
```

You should see a bunch of notifications in the terminal. After it's done, you make sure you have a dyno running to serve the app:

```
$ heroku ps:scale web=1
Scaling web processes... done, now running 1
```

Heroku gives you one free dyno to use per month. Any more than that, and you gotta pay. But for now, all you need is one.

Okay, now you've got your code setup with a Procfile and a gunicorn server, you tested the code with 'heroku local', you've pushed your code to the Heroku platform, and you started a dyno, all with just a few commands. The only thing left to do is to open it up and check it out!

```
$ heroku open
```

This will launch a browser window pointed toward your app on the Heroku servers. (If it doesn't work, you can always find your app via the Heroku Dashboard or at http://app-name.herokuapp.com .

And that's the basics of the Heroku platform. The great thing about Heroku (in addition to giving you a free dyno, which is enough for hosting a personal app), is that it's so easy to use and integrate with your existing git workflow. If you're interested in getting to know Heroku better, you can check out their Python guide . Some good things to know:

- You can run a command, typically scripts and applications that are part of your app, in a one-off dyno using the `heroku run` command.
- You can view a stream of your application logs with `heroku logs --tail` . Use Ctrl-c to exit.
- There are a ton of add-ons for Heroku that can help you manage your app: Heroku Add-ons .

- One of those add-ons is a free Postgres database, which will be helpful in any of your capstone projects that require a database. (There are a bunch of other databases available too.)
- If your app uses environment variables (for example, a DB_USER and DB_PASS in the SQL configuration), you can set the Heroku environment variables with `heroku config:set VAR_NAME=value` .

Now that you're finished with your project, you should submit the Heroku link for your project.

What is a Heroku dyno?

> *A dyno is a virtual machine on Heroku that is responsible for running some task*

What is the procfile used for? How does this relate to 'heroku local'?

> *The procfile specifies which command should be run to start a web dyno. The procfile tells 'heroku local' what to do.*

How do we tell Heroku what the dependencies for our application are?

> *Save them in a requirements.txt file. You can do this by running* `pip freeze > requirements.txt` *, or by listing them one per line.*

Can you describe the workflow for deploying changes to an application on Heroku?

> *Develop your code on a branch (call it some-branch), then run "git push heroku some-branch" when you're ready to deploy. The first time around, you'll need to run* `heroku create` *before pushing up to Heroku.*

How do you monitor the output of your application?

> *You can look at the logs by running* `heroku logs`.

What does the `heroku run` command do? Can you think of reasons why you might use this command?

> `heroku run` *allows you to run a command in a one off dyno. This can be helpful for debugging your Heroku environment or running one off tasks like a database migration or testing.*

☆ ☆ ☆ ☆ ☆ · Report a typo or other issue

example.com/project

✓ Submit your project

Previous

Next