Open in app

Moses Gitau

Follow 50 Followers About

Deploying a Flask application on Heroku

Moses Gitau May 31, 2018 · 3 min read

Andela Bootcamp day 4, and I got the chance to deploy my first application to Heroku, and I would say, it was one of the challenging issues I have faced so far. It all began when my application was ready for deployment, I did my research as usual and had a few resources helping me out in the process, watching YouTube tutorials and reading medium blogs. After hours browsing relentlessly, I managed to pick up a few crucial instructions that I will discussion in the following section.

Open in app

Installing Heroku CLI

This will help you manage your application directly from command line.

Download and Install Heroku CLI

Creating a Heroku Account

If you don't have a Heroku account yet, don't worry it's free, you will head over to Heroku and get yourself a free account.

Getting started

You will need to have basic knowledge of git commands, using terminal (command line), running virtual environments in Python and creating Flask applications.

Login to heroku in your terminal with the following command.

```
heroku login
```

You will be prompted to enter your email and password.

Open in app

Initializing a git repository

Run the following command to initialize a git repository. Skip this step if you already have a git repository.

```
git init
```

Create a heroku application

You will be required to create a heroku application. In this example, I will create an application called **your-first-heroku-app,** which will be accessed from the link: *https://your-first-heroku-app.herokuapp.com.* Feel free to name your application as you wish.

```
heroku create your-first-heroku-app --buildpack heroku/python
```

Add the remote heroku git repository. (remember to replace **your-first-heroku-app** with the name of the heroku app you chose):

```
heroku git:remote -a your-first-heroku-app
```

Install gunicorn

Gunicorn is a python WSGI HTTP server that will serve your Flask application at heroku.

Open in app

Finally you will need to create **requirements.txt** file in your project root folder in order for heroku to detect it as a Python project. Run the following command within the project root folder.

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

Ensure your requirements.txt is in the project root folder, else your heroku application will fail to deploy.

Adding a Procfile

Create a `Procfile` in the project root folder and add the following line:

```
web: gunicorn app:app
```

The first `app` represents the name of the python file that runs your application or the name of the module it is in. The second `app` represents your app name. For example if your application runs from a `run.py` file that looks like this:

```
from flask import Flask

my_awesome_app = Flask(__name__)


@my_awesome_app.route('/')
def hello_world():
    return 'Hello World!'


if __name__ == '__main__':
    my_awesome_app.run()
```

Then you will add the following line in your Procfile

Open in app

Committing the files to Heroku's repository master branch

Stage the files for a commit and commit them

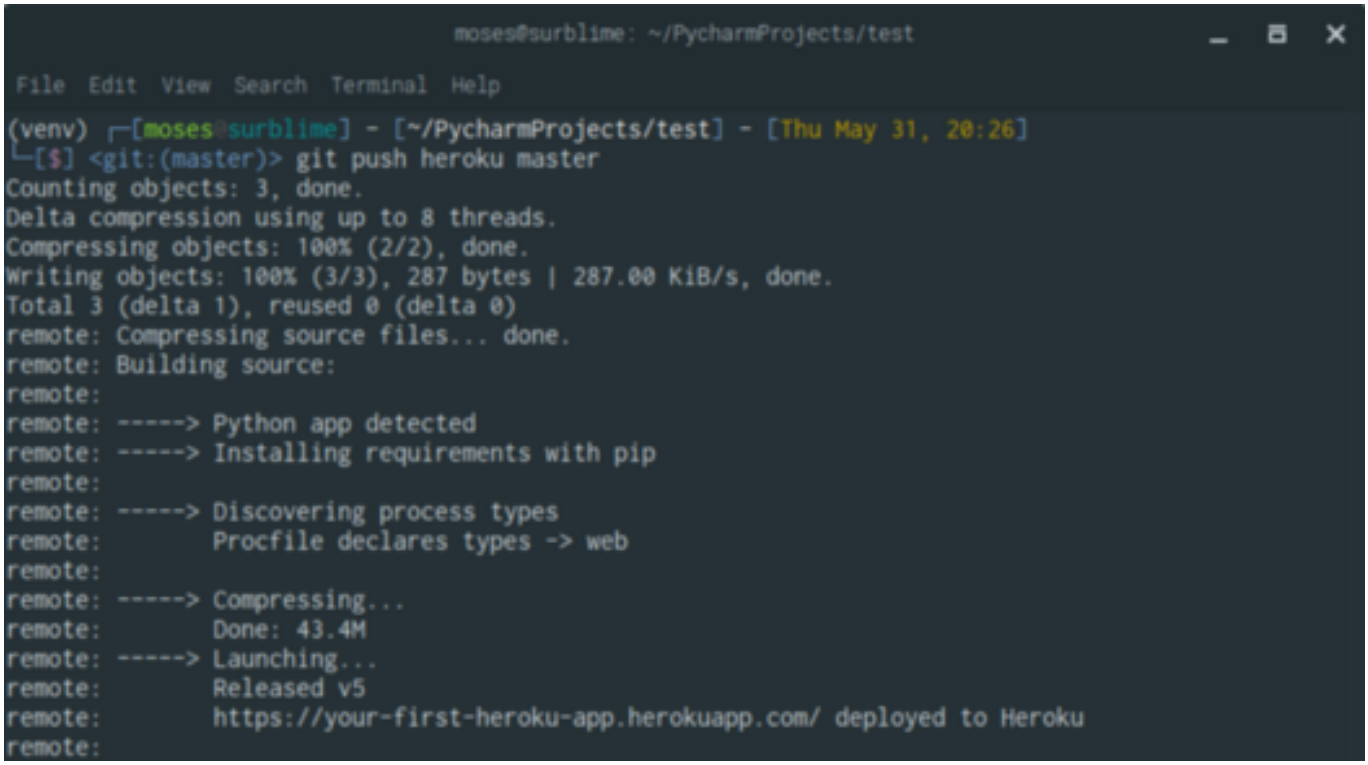
```
git add .
git commit -m "First commit for heroku"
```

Push the changes from your local `master` branch to heroku's `master` branch.

```
git push heroku master
```

If you want to push from a different branch, say `develop` branch, you can use the following command

```
git push heroku develop:master
```



Open in app

Almost there

Now fire up your browser and got to: (remember to replace **your-first-heroku-app** with the name of the app you created.)

https://your-first-heroku-app.herokuapp.com

Web Development Flask Heroku

AboutWriteHelpLegal

Get the Medium app

Download on the App Store GET IT ON Google Play