How to Deploy a React + Python Flask Project on Heroku

Steffy Lo Follow
May 17 · 4 min read ★

Setting Up For Deployment

First, you'll need to build your React project by running the following command:

```
npm run build
```

After the build process completes, a directory named "build" should appear. This is the folder that we want to deploy onto Heroku.

```
build
├── asset-manifest.json
├── favicon.ico
├── index.html
```

```
├── logo192.png
├── logo512.png
├── manifest.json
├── robots.txt
├── service-worker.js
└── static
```

Inside the folder should look something like the above, where index.html is the page we want to load as soon as the flask app starts. We can do this by adding the following route:

```
@app.route('/')
def index():
    return app.send_static_file('index.html')
```

However, this route won't work just yet. Before that, we'll need to point our static folder to the "build" directory. This can be done by passing in some arguments when creating the Flask app instance like so:

```
app = Flask(__name__, static_folder='./build', static_url_path='/')
```

- The static_folder argument tells Flask where the static folder is. By default this will be located in the same directory where the application is
- The static_url_path argument tells Flask what is the URL prefix for all static files. The default is /static. By changing it to the root URL, we now don't need to prepend every static file with /static

Finally, we want our app to run on the standard port 80.

```
if __name__ == "__main__":
    app.run(host='0.0.0.0', debug=False, port=os.environ.get('PORT',
80))
```

Installing gunicorn

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

```
pip install gunicorn
```

Note: if you don't have pip yet, you'll have to install it first.

Now, create a requirements.txt file in your project root folder using the following command:

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

Run the command within the project root folder and make sure that the generated requirements.txt file is in your project root folder. Your app will not be deployed correctly if it is not in the root folder!

Procfile

Create a Procfile in the project root folder and add the following line. when creating the file, make sure that your Procfile doesn't have any extensions like .txt

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

app represents the name of the python file that runs your application or the name of the module it is in. **name** represents your app name. If your application runs from a server.py file and looks like…

```
from flask import Flask
flask_app = Flask(__name__)

@flask_app.route('/')
def index():
    return app.send_static_file('index.html')
```

```
if __name__ == '__main__':
    flask_app.run(host='0.0.0.0', debug=False,
port=os.environ.get('PORT', 80))
```

Then you should add the following line in your Procfile instead:

```
web: gunicorn server:flask_app
```

We're now ready to deploy!

Deploying to Heroku

If you don't already have the Heroku CLI on your system, start by installing it: https://devcenter.heroku.com/articles/heroku-cli

After installing Heroku, login to Heroku from your terminal using the following command:

```
heroku login
```

Next, initialize a git repository. You can skip this step if you've already initialized one.

```
git init
```

Creating the Heroku App

Now, we will want to create a Heroku app. In the following example, we are creating a Heroku app with the name "react-flask-app". Note that your application will be accessed from the link `https://<insert-app-name>.herokuapp.com` . Go ahead and choose the best name that is available for your app.

```
heroku create react-flask-app
```

Then, add the remote heroku git repository, replacing “react-flask-app” with the name of the heroku app you chose:

```
heroku git:remote -a react-flask-app
```

Deploy! Push to Heroku Master Branch

```
git add .
git commit -m "initial commit for heroku"
git push heroku master
```

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

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

Finally, make sure that the files you want are right where you want it to be:

```
heroku run bash -a react-flask-app
$ ls
```

After double-checking, exit and return to the terminal.

```
$ exit
```

Congratulations! Your app is now deployed!

Simply go to react-flask-app.herokuapp.com (where react-flask-app is the name you chose for your app) and you should see your newly deployed app!

P.S. A personal React + Flask Project I've deployed on Heroku can be found here: https://animania-pwa.herokuapp.com

Repository Link: https://github.com/steffy-lo/animania

Feel free to check out my directory structure when you get lost. Personally, I think it's clearer to see concrete examples.

If you want to learn more about the project I've built, my friend has written a post about it which you can see here

Python Flask Heroku Software Development React