Full Stack Development
Sesi 11

DEPLOYMENT

Disclaimer

In this deployment session, we will be using Code from Sesi 9!

Deploy to Heroku

Heroku makes building and deploying applications really friendly for developers. It removes much of the burden related to building and running web applications, taking care of most infrastructure details and letting you focus on creating and improving the app. Some of the details handled by Heroku include:

- Provisioning HTTPS certificates
- Managing DNS records
- Running and maintaining servers

Heroku Account Setup

Your first step is to create a Heroku account. If you don’t have one already, you can use the Free and Hobby plan. It allows you to deploy non commercial applications, personal projects, and experiments without spending money.

You’ll be able to start using Heroku after completing the required information and confirming your email address.

Heroku Command-Line Interface (CLI)

The Heroku command-line interface (CLI) is a tool that allows you to create and manage Heroku applications from the terminal. It’s the quickest and the most convenient way to deploy your application.

You can download the CLI from here.

After installing, run the following command.

```
$ heroku login
```

This opens a website with a button to complete the login process. Click Log In to complete the authentication process and start using the Heroku CLI

Application Deployment to Heroku

The first step is to create a file named Procfile in the project’s root directory. This file tells Heroku how to run the app. You can create it by running the following command:

```
$ git init
$ echo "web: gunicorn app:connex_app" > Procfile
```

Note that this filename must start with a capital letter. This file tells Heroku to serve your application using Gunicorn, a Python Web Server Gateway Interface (WSGI) HTTP server compatible with various web frameworks, including Flask.

```
$ python3 -m pip freeze > requirements.txt
```

And add psycopg2 and gunicorn to your requirements.txt.

You can create the application in Heroku by running the following command:

```
$ heroku create hacktiv8-flask-rest (CHANGE WITH YOUR OWN NAME)
$ heroku addons:create heroku-postgresql:hobby-dev --app hacktiv8-flask-rest
```

Application Deployment to Heroku

Now after your Database has been created, get the URL of your Database using the following command,

```
$ heroku config --app hacktiv8-flask-rest
```

Now that you have got your Database URL, replace the value of app.config[‘SQLALCHEMY_DATABASE_URI’] line in the “config.py” file with this Database URL.

Because of Heroku update, URI should start with postgresql:// instead of postgres://. SQLAlchemy used to accept both, but has removed support for the postgres name.

Application Deployment to Heroku

Since you added and changed files, you need to commit them to Git. You can do this by executing the following two commands:

```
$ git add .
$ git commit -m "Add Heroku deployment files"
$ heroku git:remote -a hacktiv8-flask-rest
```

Running the above command initializes the Heroku application, creating a Git remote named heroku. Next, you can push the Git repository to this remote to trigger the building and deployment process:

```
$ git push heroku master
```

After pushing the master branch to the heroku remote, you'll see that the output displays information about the building and deployment process.
You can use the following Heroku CLI command to open your app's URL:

```
$ heroku open
```

Application Deployment to Heroku

Now the one final thing that we have to do is to create the Tables and their structure that we have defined in our code into Heroku Database.

For that, from your terminal go into the Python terminal of Heroku using,

```
$ heroku run python
```

From here, run build_database file using following command

```
$ exec(open("build_database.py").read())
```

You can access your swagger from:

https://flask-ocbc-2.herokuapp.com/api/ui/

External References

References - here