Heroku Deployment Guide

This guide provides instructions for setting up Heroku and using it to deploy applications.

Prerequisites

- Sign up for a Heroku account.

- Install the Heroku CLI tool.

Deploy an Application

In this activity, you will deploy a Pet Pals application to Heroku. The application takes the name of a pet and plots its location on the map. The code for the application is not nearly as important as the steps for deploying the application to Heroku. These steps can be repeated for any application using similar architecture (i.e. Flask and sqlite).

Before deploying the application, you will follow these steps:

1. Create a GitHub repo for the application.
2. Prepare the application with additional configuration files (`Procfile` and `requirements.txt`)
3. Create the Heroku application.
4. Prepare the Heroku database.

Part 1: Create a New Repo

- **Files:** Pet Pals app

1. In GitHub, create a new repo called `Pet_Pals` and clone it to your desktop.

2. Add the starter files to this repo.

Part 2: Create the Configuration Files

1. Create a new conda environment to use just for this app. All of your project dependencies will be installed in this environment. **Note:** This environment should only contain Python 3.6 and not Anaconda. To do so, run the following code in the terminal:

   ```
   conda create -n pet_pals_env python=3.6
   ```

2. Activate the environment by running the following code:

```
conda activate pet_pals_env
```

~~~
* **Note:** If the previous code doesn’t work, try the following command
instead:

```sh
source activate pet_pals_env
```
~~~

3. Install `gunicorn` with `pip install gunicorn`. `gunicorn` is a high-performance web server that can run the Flask app in a production environment.

4. Because this app will use Postgres, install `psycopg2` with `pip install psycopg2`.

5. Install the remaining libraries into your new environment by using the following code:

   ```
   pip install flask
   pip install flask-sqlalchemy
   pip install pandas
   ```

6. Navigate to the folder that contains `initdb.py` and run the following code to initialize the database.

   ```
   python initdb.py
   ```

7. Create a shell file and add the following code:

   ```
   FLASK_APP=pet_pals/app.py flask run
   ```

8. Test the application by running the following in the terminal:

   ```
   ./run.sh
   ```

9. Navigate to `127.0.0.1:5000` to view your webpage.

10. Now that all of the project dependencies are installed, generate the `requirements.txt` file. This file is a list of the Python packages required to run the app. Heroku will use this file to install all of the app's dependencies. To do so, run the following code:

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

11. Create the final configuration file, which is `Procfile`. This file is used by Heroku to run the app. To do so, run the following code:

```
touch Procfile
```

12. Add the following code to the `Procfile` which instructs Heroku how to run the app.

```
web: gunicorn pet_pals.app:app
```

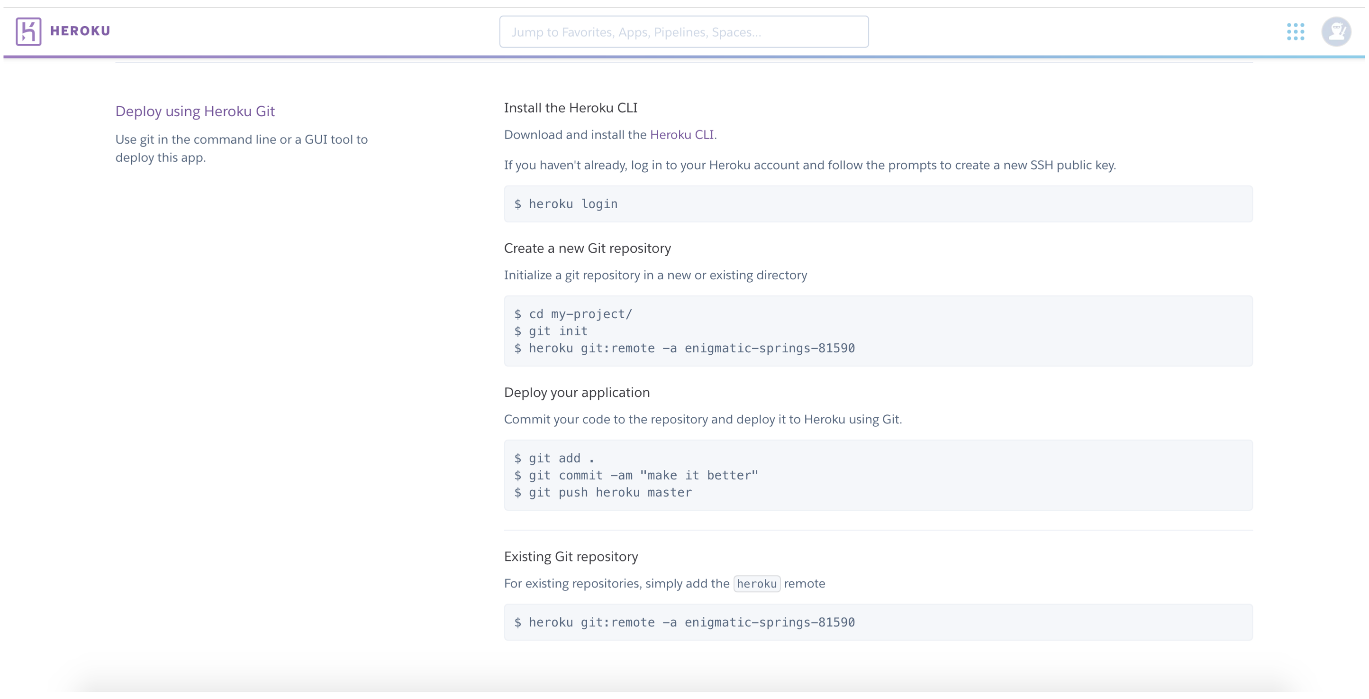
Note: `pet_pals` is the name of the folder that contains your app as a python package (i.e., the name of the folder with the `__init__.py` file in it).

13. Add, commit, and push the configuration and starter filesto your repo.

Part 3: Create the Heroku App

1. On the main dashboard, click New (located in the type right of the dashboard) and select **Create a new app**. Then, give your app an unique name and keep the default region.

2.Go to the Deploy section of your app's homepage, and follow the steps to deploy the app.



Deploy using Heroku Git

Use git in the command line or a GUI tool to deploy this app.

Install the Heroku CLI

Download and install the Heroku CLI.

If you haven't already, log in to your Heroku account and follow the prompts to create a new SSH public key.

```
$ heroku login
```

Create a new Git repository

Initialize a git repository in a new or existing directory

```
$ cd my-project/
$ git init
$ heroku git:remote -a enigmatic-springs-81590
```

Deploy your application

Commit your code to the repository and deploy it to Heroku using Git.

```
$ git add .
$ git commit -am "make it better"
$ git push heroku master
```

Existing Git repository

For existing repositories, simply add the `heroku` remote

```
$ heroku git:remote -a enigmatic-springs-81590
```

3. In the "Deployment method" section, select GitHub.

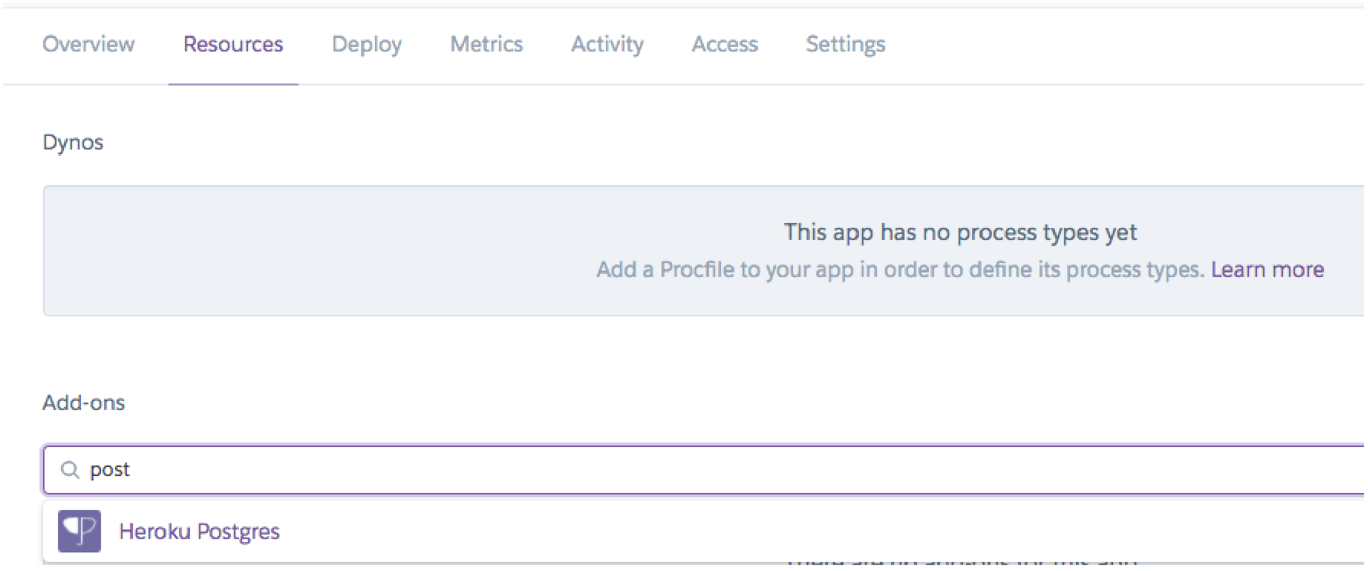
4. Once connected to GitHub, search for the `pet-pals` repo you created that contains your code from part 2 of this guide, and connect.

5. With your repo selected, navigate to the "Manual deploy" section and click **Deploy Branch**.

6. To confirm that your app has been successfully deployed, navigate to the top of the page and click **Open app**. This should open a webpage with your pet pals webpage. **Note:** The database has not been set up yet, so there is one more step before it is fully functioning.

Part 4: Prepare the Postgres Database

1. After creating a new app on Heroku, navigate to the Resources tab in the menu bar underneath the name of your app:



Dynos

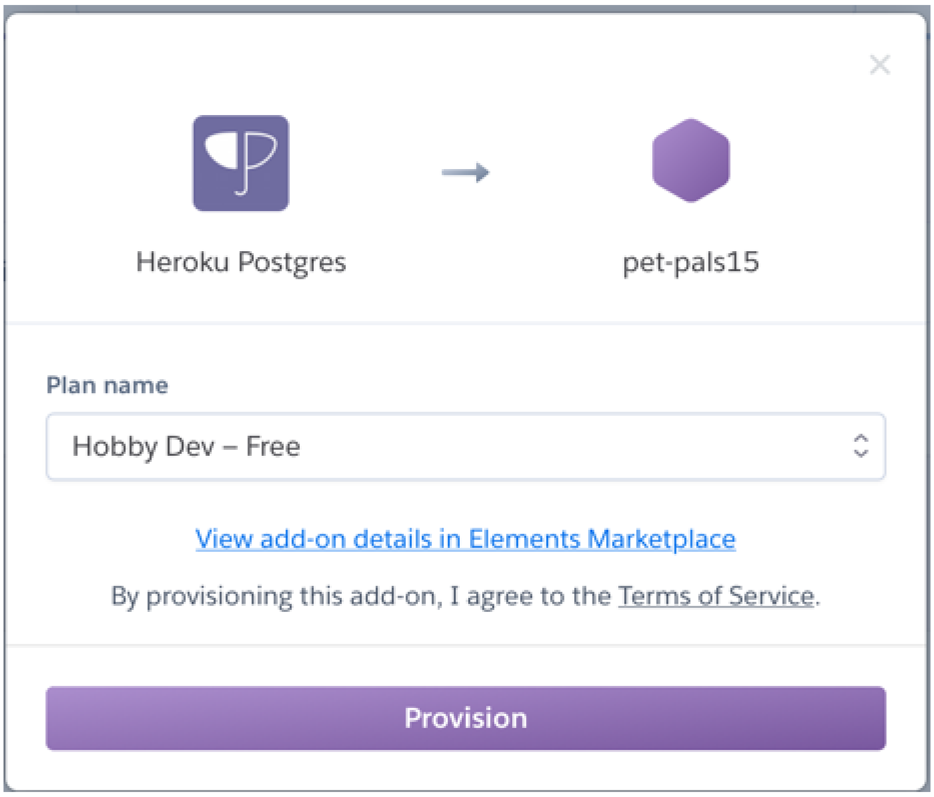
This app has no process types yet
Add a Procfile to your app in order to define its process types. Learn more

Add-ons

post

Heroku Postgres

2. Under "Add-ons," search "Heroku Postgres." Make sure to select the free version, and then click Provision.



Heroku Postgres → pet-pals15

Plan name

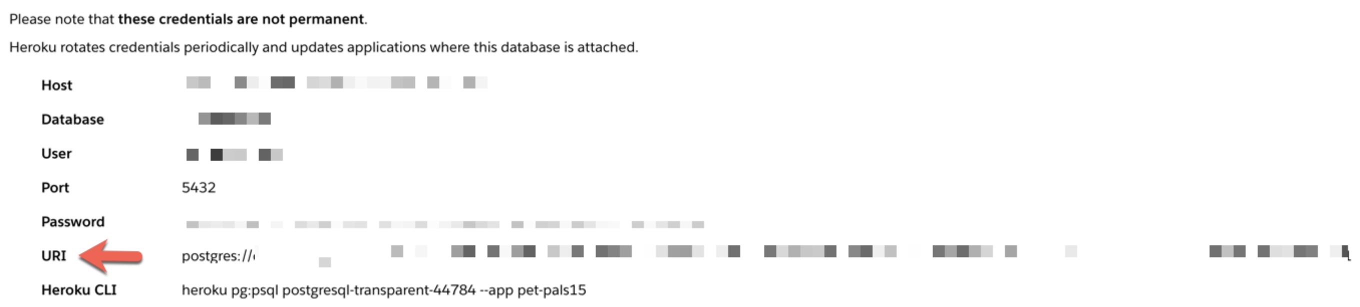
Hobby Dev – Free

View add-on details in Elements Marketplace

By provisioning this add-on, I agree to the Terms of Service.

```
* Once Heroku Postgres is listed, click it.
```

3. From the new page, navigate to Settings and click **View Credentials**. The connection string to the database should now be available in the URI field:


Please note that **these credentials are not permanent**.

Heroku rotates credentials periodically and updates applications where this database is attached.

Host	[redacted]
Database	[redacted]
User	[redacted]
Port	5432
Password	[redacted]
URI	postgres://[redacted]
Heroku CLI	heroku pg:psql postgresql-transparent-44784 --app pet-pals15

~~~~
* **Note:** Heroku automatically assigns this URI string to the
`DATABASE_URL` environment variable that is used in `app.py`. The code
that follows is already in `app.py`, and is what uses that environment
variable to connect to the Heroku database.

  ```python
  # DATABASE_URL will contain the database connection string:
  app.config['SQLALCHEMY_DATABASE_URI'] = os.environ.get('DATABASE_URL',
'')
  # Connects to the database using the app config
  db = SQLAlchemy(app)
  ```
~~~~

Note: The final step requires the Heroku CLI. If you do not currently have it installed, follow the instructions for installing the Heroku CLI.

4. After adding the database, the final step is to initialize the database by using the Heroku CLI. Run the following code, replacing `<name of your app>` with the name of your app as it appears in Heroku:

   ```
   heroku run python initdb.py -a <name of your app>
   ```

Congratulations! Your database is now initialized, and you can open the application by using `heroku open` from the terminal or from **Open App** on the webpage.

- If you plan on using MongoDB as a data source, the following links can serve as a starting point:
- Deploy to Heroku with Mongo
- How to deploy your Mongo app