

Heroku Deployment

Instructions

- In this activity, we will deploy the Pet Pals application to Heroku. This step consists of 3 main parts:
 1. Prepare the application with additional configuration files (`Procfile` and `requirements.txt`)
 2. Create the Heroku application
 3. Prepare the Heroku database

Part 1: Configuration Files

- If you haven't already, send the code from the previous activity to the class.
- Start by creating a new conda environment just for this app. All of our project dependencies will be installed in this environment. Note: This should only contain python 3.6 and not anaconda.

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

- Make sure to activate this new environment before proceeding.

```
source activate pet_pals_env
```

- Next, we install `gunicorn` with `pip install gunicorn`. Explain that gunicorn is a high performance web server that can run their Flask app in a production environment.
- Because this app will use Postgres, we also install `psycopg2` with `pip install psycopg2`.
- Make sure to install any other dependencies that are required by the application. This may be `Pandas`, `flask-sqlalchemy`, or any other Python package that is required to run the app. **Test the app locally to make sure that it works!**

```
pip install gunicorn
pip install psycopg2
pip install flask
pip install flask-sqlalchemy
pip install pandas
```

- Test the app by first initializing the database:

```
python initdb.py
```

- Run the app using the following:

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

- Now that all of the the project dependencies are installed, we need to generate the `requirements.txt` file. This file is a list of the Python packages required to run the app, we run `pip freeze > requirements.txt`. Heroku will use this file to install all of the app's dependencies.
- The final configuration file that we need is `Procfile`. This file is used by Heroku to run the app.

```
touch Procfile
```

- Open Procfile in vscode and add the following line:

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

- Explain that `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).

Part 2: Creating the Heroku App

- On Heroku, go to the `Deploy` section of your app's homepage, and follow the steps to deploy the app.

The screenshot shows the Heroku CLI guide for deploying an application using Git. The guide is divided into several sections with instructions and terminal commands.

HEROKU Jump to Favorites, Apps, Pipelines, Spaces...

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
```

Part 3: Preparing the Database

- After creating a new app on Heroku, navigate to `Resources` :

Overview Resources Deploy Metrics Activity Access Settings

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

- Under **Add-ons**, add **Heroku Postgres**. Make sure to use the free version.
 - Click on the add on, then navigate to settings and click on **Reveal Config Variables**.
 - The connection string to the database should now be available:

HEROKU Jump to Favorites, Apps, Pipelines, Spaces...

Personal > enigmatic-springs-81590 Open app More

Overview Resources Deploy Metrics Activity Access Settings

Name enigmatic-springs-81590 Edit

DATABASE_URL contains the database connection string that the app will use to connect to the database.

Config Variables

Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.

Config Vars

KEY	VALUE
DATABASE_URL	postgres://lgpvhiavfjqmvp:a0fff162c8d...

Hide Config Vars Add

- Heroku will automatically assign this URI string to the **DATABASE_URL** environment variable that is used within **app.py**. The code that is already in **app.py** will be able to use that environment variable to connect to the Heroku database.

```
# 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)
```

- After adding the database, the final step is to initialize the database. To do this, we use the heroku cli. From the terminal, type the following:

```
heroku run initdb.py
```

- Your database is now initialized, and you can open the application using `heroku open` from the terminal.