Django 9: Deployment

IN608: Intermediate Application Development Concepts

Kaiako: Tom Clark & Grayson Orr

Last Session’s Content

- Django REST framework
  - Permissions
  - Serialization
  - Viewsets
  - Routers

Today’s Content

- Heroku deployment
  - Heroku CLI
- Heroku Postgres
- Deploy Django application

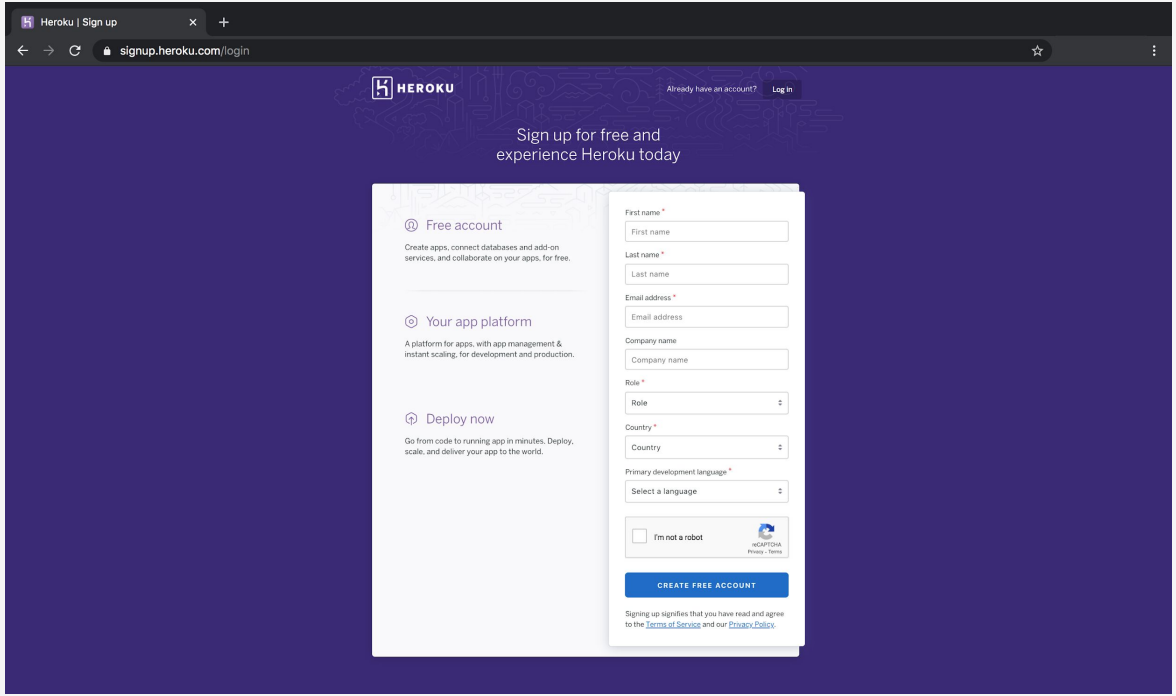
Heroku Deployment

Heroku Deployment

- Heroku lets you build, deploy & manage applications written in a variety of Ruby, Node.js, Java, Python, Clojure, Scala, Go & PHP
- An application's source code & dependency file, i.e., `Pipfile.lock` should provide enough information for the Heroku platform to build your application
- How does Heroku know what command(s) to run?
  - If you are using a framework, i.e., Django or Ruby on Rails, Heroku can figure out what command(s) to run, i.e., `python manage.py runserver` or `rails server`
- You may need to explicitly declare what can be executed. To do this, you need to create a `Procfile`. We will look at this soon
- Each line in `Procfile` declares a process type which is a command that can be executed against an application

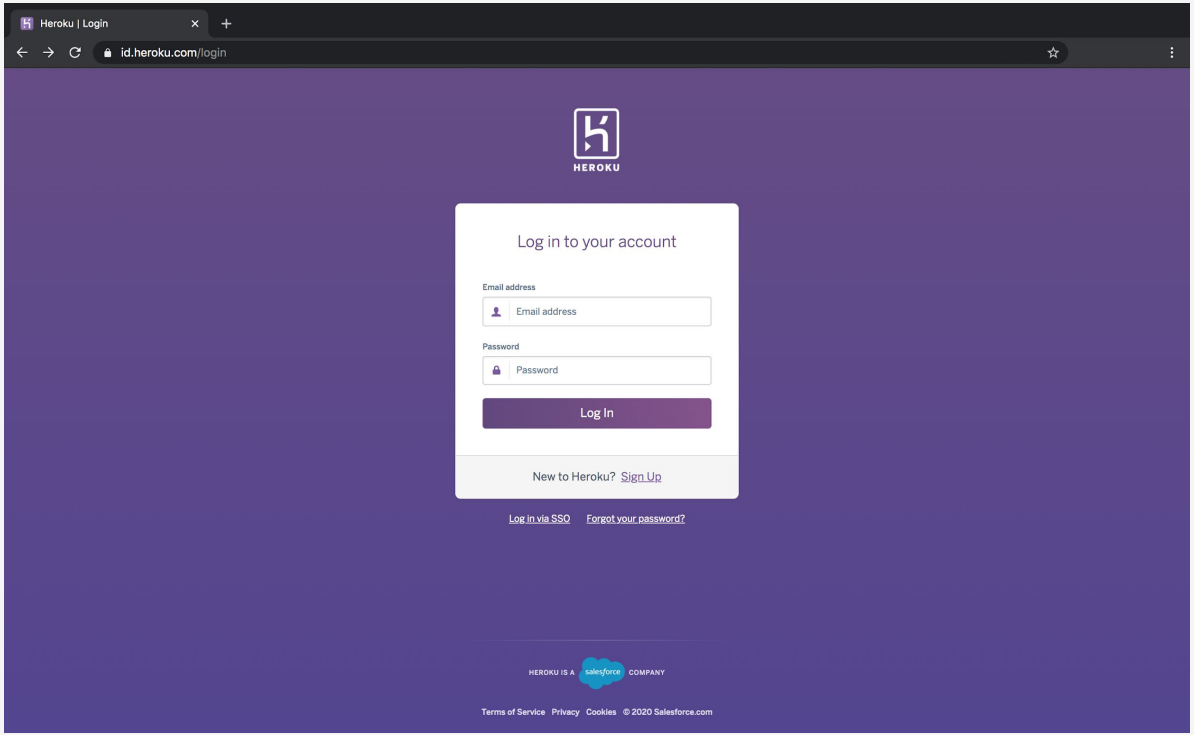
Heroku Deployment

- Sign up to Heroku



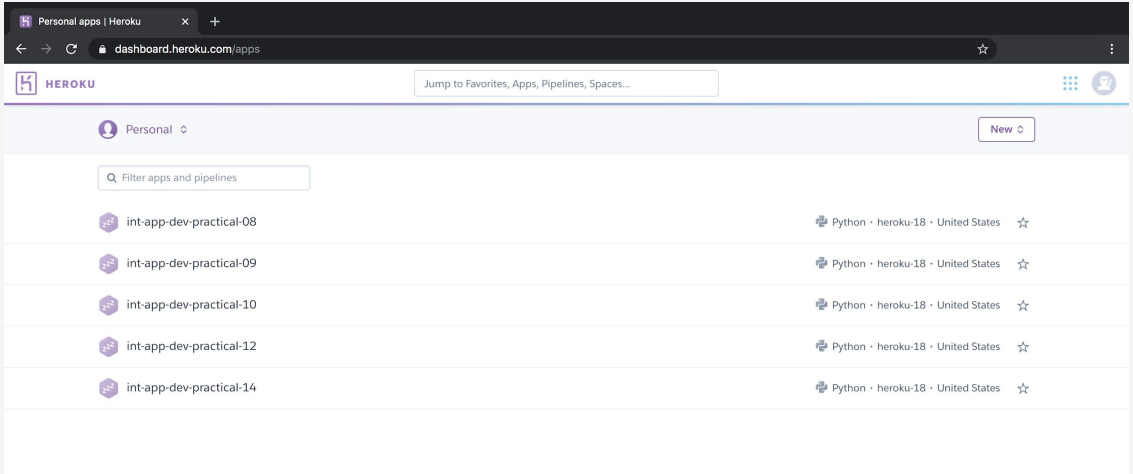
Heroku Deployment

- Login to Heroku



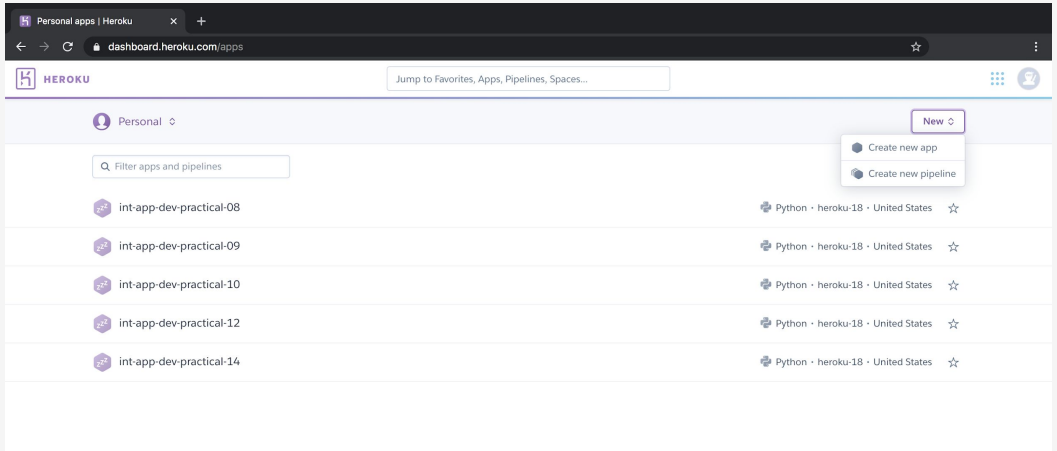
Heroku Deployment

- Heroku dashboard
- Displays all applications



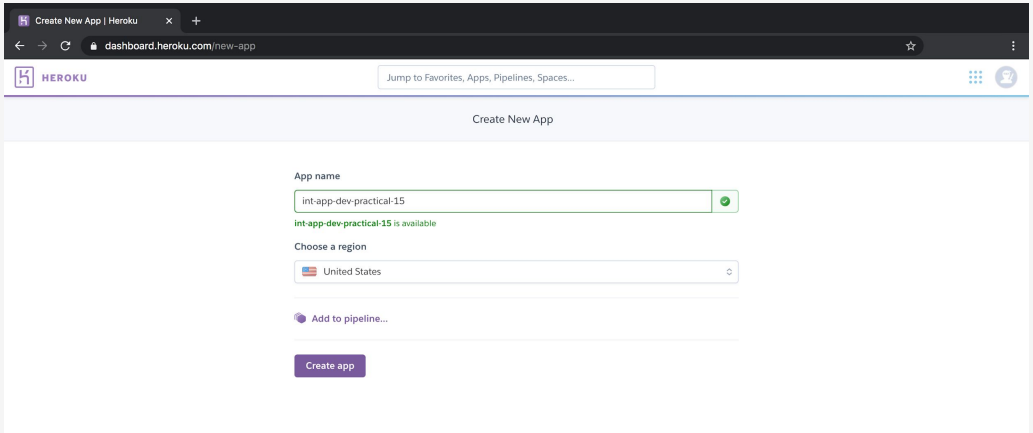
Heroku Deployment

- New > Create new app
- You can also create a new application using Heroku CLI



Heroku Deployment

- Create a new application via GUI



Heroku Deployment

int-app-dev-practical-15 · Her

dashboard.heroku.com/apps/int-app-dev-practical-15/deploy/heroku-git

HEROKU

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

Personal > int-app-dev-practical-15

Open app | More

Overview | Resources | Deploy | Metrics | Activity | Access | Settings

Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features

Pipelines let you connect multiple apps together and **promote code** between them. Learn more.

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. Learn more.

Choose a pipeline

Deployment method

Heroku Git — Use Heroku CLI

GitHub — Connect to GitHub

Container Registry — Use Heroku CLI

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 int-app-dev-practical-15
```

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
```

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.

Existing Git repository

For existing repositories, simply add the `heroku` remote

```
$ heroku git:remote -a int-app-dev-practical-15
```

Heroku CLI

- Heroku CLI (Command Line Interface) makes it easy to build, deploy & manage your Heroku applications from the terminal
- Heroku CLI requires Git
- Installation available for macOS, Windows & Linux
  - The Window installer may display a warning - “Windows protected your PC”
  - To run the installation, click “More info”, verify the publisher as “Heroku, Inc.”, then click the “Run anyway” button
- To verify your installation, run the command: `heroku --version`
  - In the output, you should see `heroku/x.x.x`
- After you install Heroku CLI, run the command: `heroku login`
  - You will be prompt to enter any key. This will navigate you to your web browser to complete your login
  - Heroku CLI will then automatically log you in
- Resource: https://devcenter.heroku.com/articles/heroku-cli

Heroku CLI

- We created an application using the GUI...how do we create an application using the CLI?
  - Create an application without a name - `heroku create`. **Note:** a random name will be generated
  - Create an application with a name - `heroku create int-app-dev-practical-15`
- We are going to use the `int-app-dev-practical-15` application as an example
- Initialise a git repository in a new or existing directory, i.e., `git init`
- If you have an existing application on Heroku, i.e., `int-app-dev-practical-15`, you can add a remote to your local repository by running the command: `heroku git:remote -a int-app-dev-practical-15`
- To deploy the application:
  - Add changes in the working directory to the staging area - `git add .`
  - Capture the state of the project at that point of time - `git commit -m “<some message>”`
  - Upload content in the local repository to the remote repository - `git push heroku master`
- To view your application, run the command: `heroku open`

Heroku CLI

- To run our Heroku application locally, run the command: `heroku local:start`
- Navigate http://0.0.0.0:5000/practical15heroku

Heroku CLI

- An application's logs are collected from output streams of all of its running processes, system components & backing services
- Two types of logs:
  - Runtime logs, i.e., application, system, API & add-on logs
  - Build logs - separate from runtime logs while building & deploying your application
- There are various ways to view your logs:
  - `heroku logs`
  - `heroku logs --num 200`
    - By default, the `logs` command retrieves 100 log entries
    - You can specify the number of log entries to retrieve by using the `--num` or `-n` flag
  - `heroku logs --tail`
    - Displays recent logs & leaves the session open for real-time logs to stream in
    - Better insights into the behaviour of your application & debugging
- Resource: https://devcenter.heroku.com/articles/logging

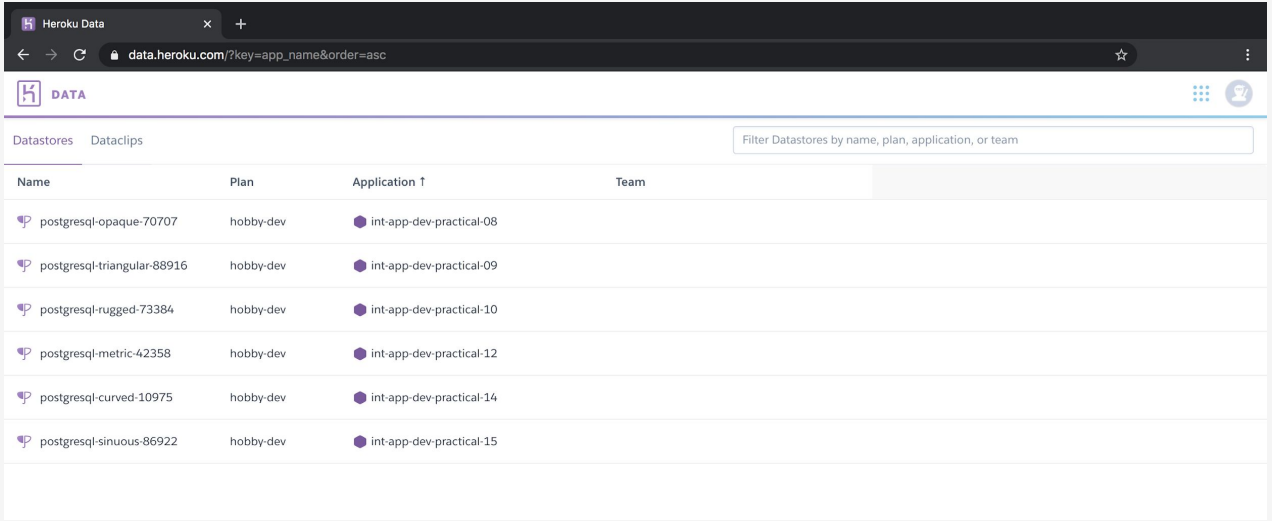
Heroku PostgreSQL

Heroku Postgres

- Heroku Postgres is a managed SQL database service provided directly by Heroku
- Accessible from any language with a PostgreSQL driver
- To check if Heroku Postgres has been provided, run the command: `heroku addons`
- If `heroku-postgresql` is not in the application's list of add-ons, run the command: `heroku addons:create heroku-postgresql:hobby:dev`
- If you want to specify a version of PostgreSQL, include the `--version` flag, i.e., `heroku addons:create heroku-postgresql:hobby:dev --version=12.4`
- We will need to apply migrations - `heroku run python manage.py migrate`
- Resource: https://devcenter.heroku.com/articles/heroku-postgresql

Heroku Postgres

- List of datastores



Heroku Postgres

postgresql-sinuous-86922 | H

data.heroku.com/datastores/1380143b-544b-4b52-afbd-cf616ab17f1f

DATA

Datastores > postgresql-sinuous-86922

SERVICE heroku-postgresql PLAN hobby-dev BILLING APP int-app-dev-practical-15

Overview Durability Settings Dataclips

HEALTH

Available

PRIMARY Yes VERSION 12.4 CREATED 13 hours ago MAINTENANCE Unsupported ROLLBACK Unsupported

UTILIZATION

0 of 20	0 of 10,000	8.1 MB	0
CONNECTIONS	ROWS IN COMPLIANCE	DATA SIZE	TABLES

Heroku Postgres

postgresql-sinuous-86922 | H

data.heroku.com/datastores/1380143b-544b-4b52-afbd-cf616ab17f1f#administration

DATA

Datastores > postgresql-sinuous-86922

SERVICE heroku-postgresql PLAN hobby-dev BILLING APP int-app-dev-practical-15

Overview Durability Settings Dataclips

ADMINISTRATION

Database Credentials

Get credentials for manual connections to this database.

View Credentials...

Reset Database

Reset the database to its originally-provisioned state, deleting all data inside it.

Reset Database...

Destroy Database

Destroys the database and all of the data inside it.

Destroy Database...

Heroku Postgres

ADMINISTRATION

Database Credentials

Get credentials for manual connections to this database.

Cancel

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

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

Host	ec2-54-91-178-234.compute-1.amazonaws.com
Database	d8ka852e4ab2fu
User	
Port	5432
Password	
URI	postgres://fivgqmxfqetjwr:063fa9703ff3ef39181d1c76f220f2596fa1a28fd399e88baa7c47d244be8d1f@ec2-54-91-178-234.compute-1.amazonaws.com:5432/d8ka852e4ab2fu
Heroku CLI	heroku pg:psql postgresql-sinuous-86922 --app int-app-dev-practical-15

Deploy Django Application

Deploy Django Application

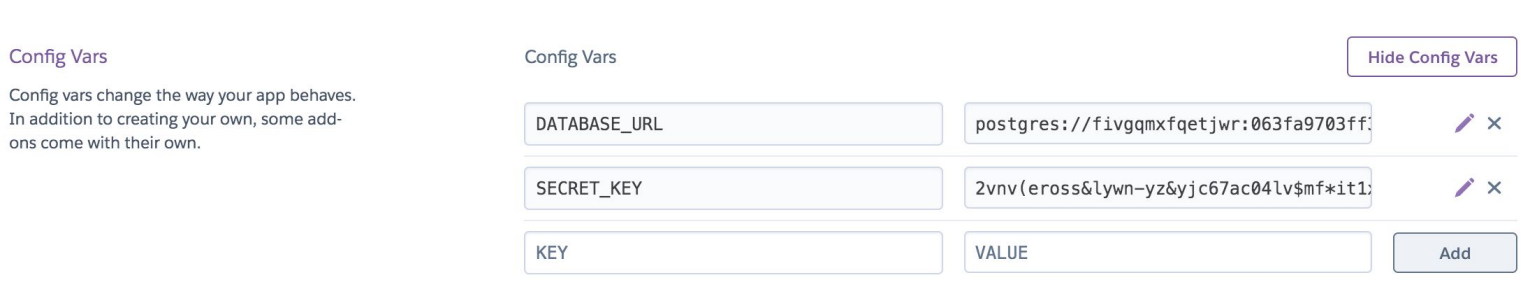
- How do we deploy a Django application?
  - Create a `.env` file for local development
  - Create a config var on Heroku
  - Configure database in `settings.py`
  - Declare the `STATIC_ROOT` configuration in `settings.py`
  - Add `WhiteNoiseMiddleware` to the `MIDDLEWARE` configuration
  - Create a `Procfile`

Deploy Django Application

- A single application always runs in multiple environments, i.e., development & production
- These environments all may run the same code, but usually have environment-specific configurations
- Environment-specific configurations should be stored in environment variables & not in the application's source code, i.e., `SECRET_KEY` in `settings.py`
- Why should we do this?
  - Modify each environment's configuration in insolation
  - Prevents credentials, i.e., database username & password from being stored in version control
- In the root directory of your Django project, create a file called `.env`
  - Used for local development
  - Do not store in version control
- Set `SECRET_KEY` in `.env`, i.e., `SECRET_KEY=<some value>`
- To use environment variables locally:
  - Install the `dotenv` Python module by running the command `pipenv install dotenv`
  - In settings.py, declare the following:
    - `from dotenv import load_dotenv`
    - `load_dotenv()`
    - `SECRET_KEY = os.environ.get('SECRET_KEY')`

Deploy Django Application

- Navigate to https://dashboard.heroku.com/apps/int-app-dev-practical-15/settings
- Heroku sets environments variables using config vars
- `DATABASE_URL` config variable is automatically created when Heroku Postgres is added on
- Set `SECRET_KEY` in config vars
- Resource: https://devcenter.heroku.com/articles/config-vars



Deploy Django Application

- How do we connect to Heroku Postgres?
  - Install the `dj-django-url` Python module by running the command `pipenv install dj-django-url`
  - In settings.py, declare the following:
    - ```
      from dj_database_url import config
      db_from_env = config(conn_max_age=600)
      DATABASES['default'].update(db_from_env)
      ```
- What is happening?
  - Converts the `DATABASE_URL` config var from Heroku into a Python dictionary
  - The dictionary is injected into the `DATABASES` configuration in `settings.py`
  - We do not have to explicitly set up Heroku Postgres in `settings.py`
      ```

Deploy Django Application

- In Django, static assets can be difficult to configure & debug
- Django does not automatically create `STATIC_ROOT` - the directory in which `collectstatic` uses
- You will need to create this directory so it will be available when `collectstatic` is run
- **Note:** Git does not support empty directories - you will need to create at least one file
- In `settings.py`, declare the following:

```
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
```

Deploy Django Application

- Django does not support serving static files in production
- `White Noise` is a Python module designed with the purpose of serving static files in production
- To install `White Noise`, run the command: `pipenv install whitenoise`
- Add `whitenoise.middleware.WhiteNoiseMiddleware` to the `MIDDLEWARE` configuration

Deploy Django Application

- Heroku applications require a `Procfile`
- Explicitly declares an application's process types & entry points
- In the root directory of your Django project, create a file called `Procfile`
- In `Procfile`, declare the following: `web: gunicorn<Django project name>.wsgi`
- This `Procfile` requires `Gunicorn` - recommended production web server for Django applications
- To install `Gunicorn`, run the command: `pipenv install gunicorn`

Programming Activity

- Checkout to master - `git checkout master`
- Create a new branch called 15-practical - `git checkout -b 15-practical`
- **Task:** Deploy your dog Django project from **Practical 10 Django 4: Template Inheritance, Static Files & CDNs** to Heroku