

# 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
  - Heroku CLI
- Postgres on Heroku
- Deploy a Django application to Heroku

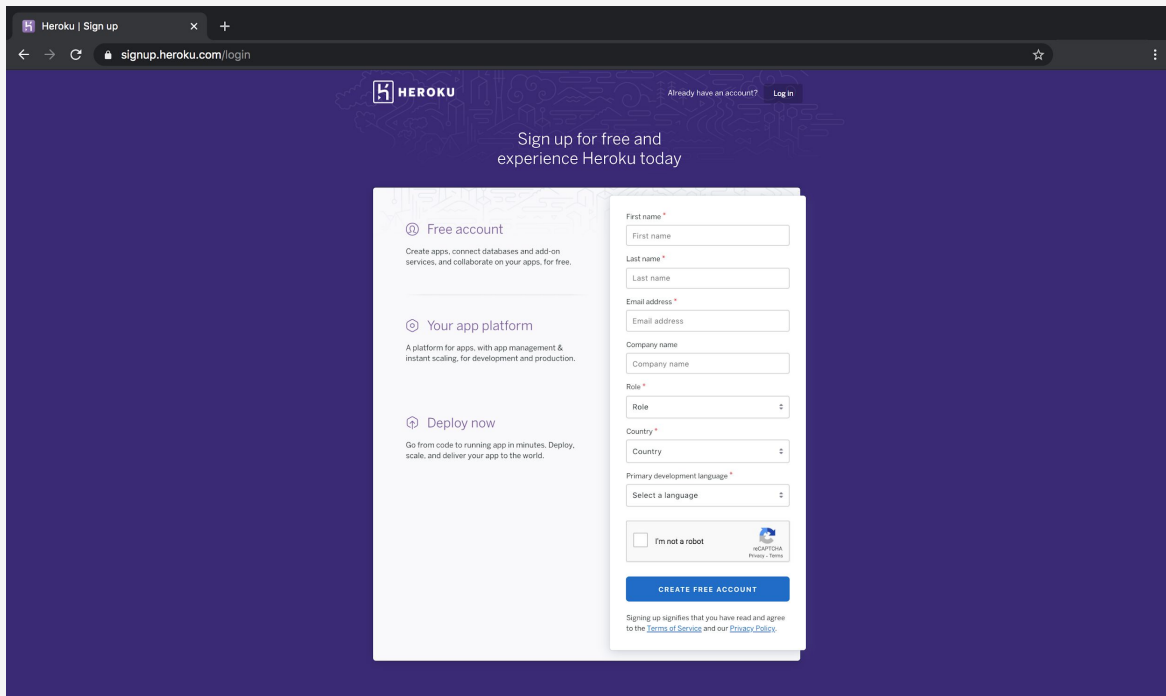
# Heroku Deployment

# 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

# Heroku Signup

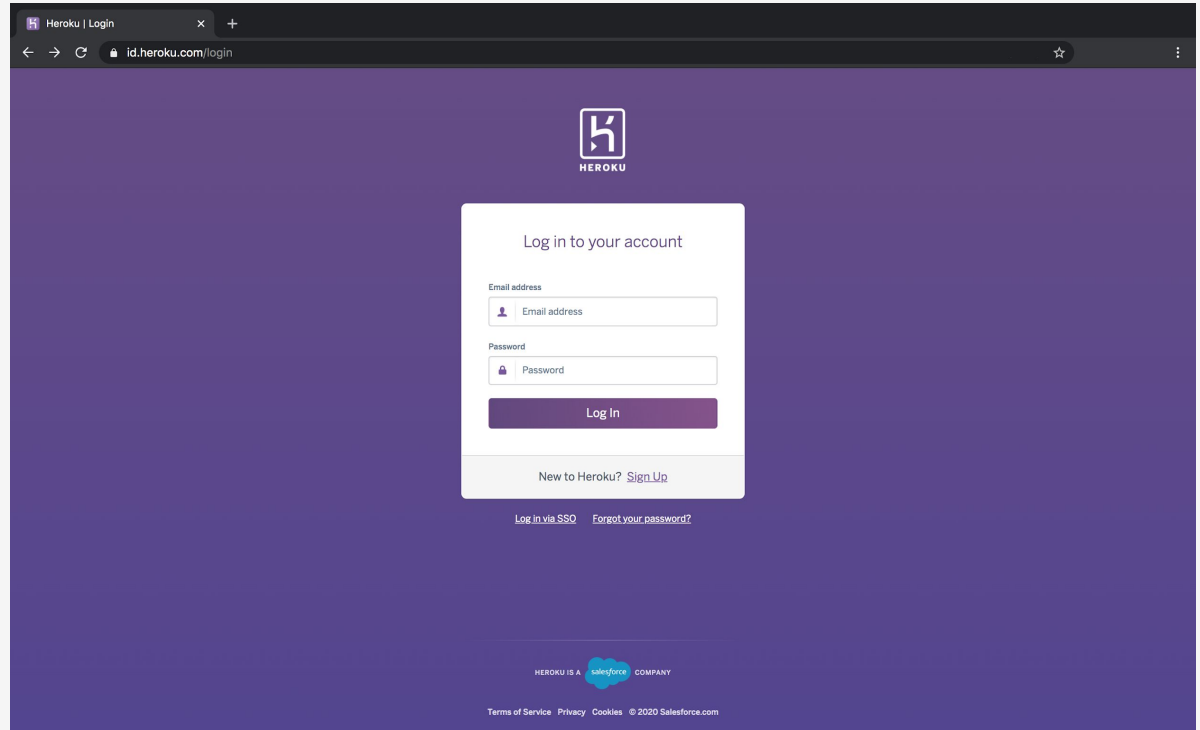
- Sign up to Heroku:  
<https://www.heroku.com/>




The screenshot shows the Heroku signup page in a web browser. The browser's address bar displays "signup.heroku.com/login". The page has a dark purple background with the Heroku logo and the text "Sign up for free and experience Heroku today". On the left, there are three sections: "Free account" (with a plus icon), "Your app platform" (with a plus icon), and "Deploy now" (with a plus icon). Each section has a brief description of the service. On the right, there is a white form with the following fields: "First name" (text input), "Last name" (text input), "Email address" (text input), "Company name" (text input), "Role" (dropdown menu), "Country" (dropdown menu), and "Primary development language" (dropdown menu). Below these fields is a checkbox labeled "I'm not a robot" and a CAPTCHA image. At the bottom of the form is a blue button labeled "CREATE FREE ACCOUNT". Below the button, there is a small disclaimer: "Signing up signifies that you have read and agree to the [Terms of Service](#) and our [Privacy Policy](#)."

# Heroku Signup

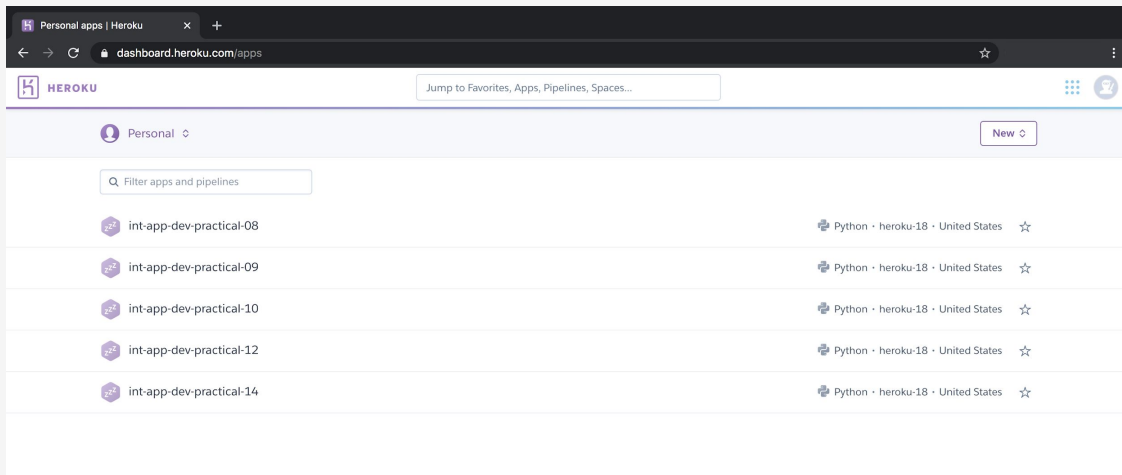
- Login to Heroku



The screenshot shows a web browser window with the address bar displaying "id.heroku.com/login". The page has a purple gradient background. At the top center is the Heroku logo, which consists of a stylized 'H' inside a square with the word "HEROKU" below it. Below the logo is a white login form. The form has the heading "Log in to your account". It contains two input fields: "Email address" with a person icon and "Password" with a lock icon. Below these fields is a purple "Log In" button. At the bottom of the form, it says "New to Heroku? [Sign Up](#)". Below the form, there are two links: "Log in via SSO" and "Forgot your password?". At the very bottom of the page, it says "HEROKU IS A  COMPANY" and "Terms of Service Privacy Cookies © 2020 Salesforce.com".

# Heroku Dashboard

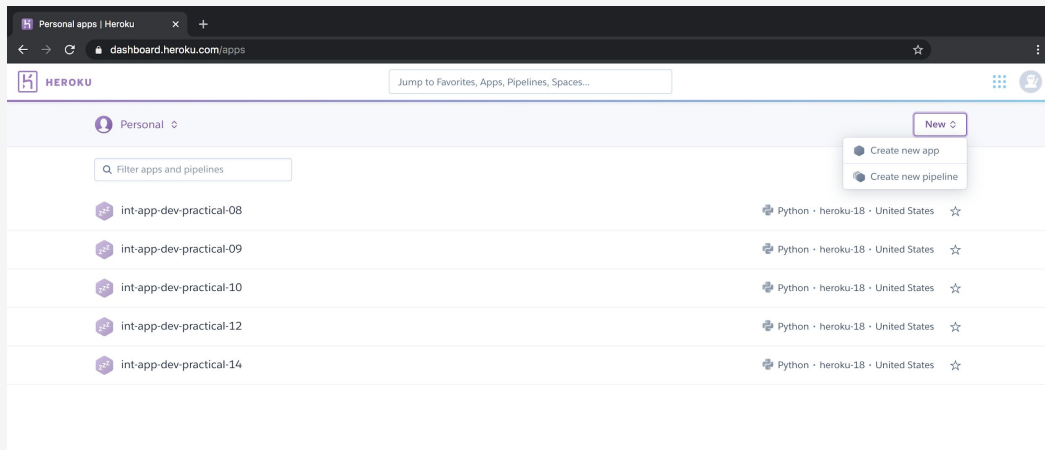
- Displays all applications





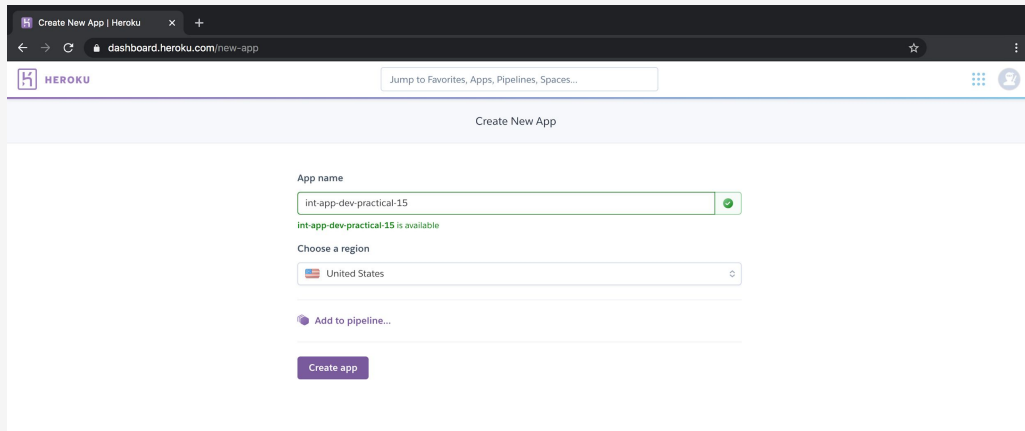
# Create an App on Heroku

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



# Create an App on Heroku

- Create a new application via GUI



# Create an App on Heroku

The screenshot shows the Heroku dashboard for an application named "int-app-dev-practical-15". The browser address bar shows the URL "dashboard.heroku.com/apps/int-app-dev-practical-15/deploy/heroku-git". The dashboard has a top navigation bar with "Personal" and the app name, and a secondary bar with tabs for "Overview", "Resources", "Deploy", "Metrics", "Activity", "Access", and "Settings".

The main content area is divided into two columns. The left column contains a section "Add this app to a pipeline" with instructions to create a new pipeline or choose an existing one. The right column contains a section "Add this app to a stage in a pipeline to enable additional features" with information about connecting multiple apps and GitHub, and a "Choose a pipeline" dropdown menu.

Below these sections is a "Deployment method" section with three options: "Heroku Git" (selected), "GitHub", and "Container Registry".

The "Deploy using Heroku Git" section provides instructions and terminal commands:

- 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 -m "make it better"
$ git push heroku master
```

A note states: "You can now change your main deploy branch from 'master' to 'main' for both manual and automatic deploys, please follow the [instructions here](#)".

The **Existing Git repository** section provides instructions 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
- Installation available for macOS, Windows & Linux
  - The Windows 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 prompted 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>

# Create an App with the Heroku CLI

- Suppose we have been developing a Django app and using Git. From the project's root directory, use the following commands:
- Create an application without a name `heroku create`. A random name will be generated
- Add a remote to your local repository by running the command:  
`heroku git:remote -a <heroku project name>`
- 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`

# Local Heroku app

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

# Heroku Logs

- 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` (Retrieves 100 recent log entries)
  - `heroku logs --num 200`
  - `heroku logs --tail` (Displays recent logs & leaves the session open)
- Resource: <https://devcenter.heroku.com/articles/logging>

# PostgreSQL on Heroku

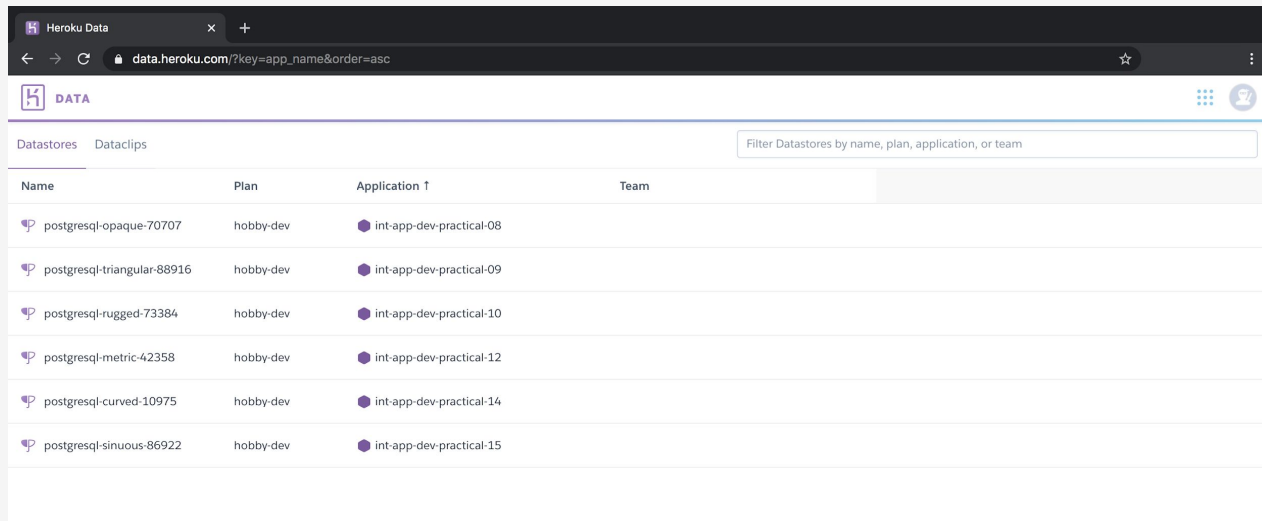


# Postgres on Heroku

- 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`
- We will need to apply migrations - `heroku run python manage.py migrate`
- Resource: <https://devcenter.heroku.com/articles/heroku-postgresql>

# Postgres on Heroku

- List of datastores



The screenshot shows the Heroku Data dashboard in a web browser. The browser's address bar displays the URL `data.heroku.com/?key=app_name&order=asc`. The dashboard header includes the Heroku logo and the word "DATA". Below the header, there are tabs for "Datastores" and "Dataclips", with "Datastores" being the active tab. A search bar on the right side of the dashboard is labeled "Filter Datastores by name, plan, application, or team". The main content area displays a table with the following columns: "Name", "Plan", "Application ↑", and "Team". The table lists six datastores, all of which are PostgreSQL databases on the "hobby-dev" plan, associated with the application "int-app-dev-practical-08" through "int-app-dev-practical-15".

Name	Plan	Application ↑	Team
postgresql-opaque-70707	hobby-dev	int-app-dev-practical-08	
postgresql-triangular-88916	hobby-dev	int-app-dev-practical-09	
postgresql-rugged-73384	hobby-dev	int-app-dev-practical-10	
postgresql-metric-42358	hobby-dev	int-app-dev-practical-12	
postgresql-curved-10975	hobby-dev	int-app-dev-practical-14	
postgresql-sinuous-86922	hobby-dev	int-app-dev-practical-15	

# Postgres on Heroku

The screenshot shows the Heroku Datastores interface for a PostgreSQL instance named 'postgresql-sinuous-86922'. The browser address bar shows the URL 'data.heroku.com/datastores/1380143b-544b-4b52-afbd-cf616ab17f1f'. The page header includes the Heroku logo and 'DATA' label. The breadcrumb trail is 'Datastores > postgresql-sinuous-86922'. Below this, there are links for 'SERVICE heroku-postgresql', 'PLAN hobby-dev', 'BILLING APP', and 'int-app-dev-practical-15'. The 'Overview' tab is selected, with other tabs being 'Durability', 'Settings', and 'Dataclips'. The 'HEALTH' section shows a green checkmark and 'Available'. Below this, a row of metadata includes 'PRIMARY Yes', 'VERSION 12.4', 'CREATED 13 hours ago', 'MAINTENANCE Unsupported', and 'ROLLBACK Unsupported'. The 'UTILIZATION' section displays four metrics: '0 of 20' for connections, '0 of 10,000' for rows (with a green 'IN COMPLIANCE' status), '8.1 MB' for data size, and '0' for tables.

postgresql-sinuous-86922 | H x +

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 <span>✓ IN COMPLIANCE</span>	DATA SIZE	TABLES

# Postgres on Heroku

The screenshot shows the Heroku PostgreSQL administration interface in a web browser. The browser's address bar displays the URL `data.heroku.com/datastores/1380143b-544b-4b52-afbd-cf616ab171f1#administration`. The page header includes the Heroku logo and the word "DATA". Below the header, a breadcrumb trail shows "Datastores > postgresql-sinuous-86922". A navigation bar lists "SERVICE heroku-postgresql", "PLAN hobby-dev", "BILLING APP", and "int-app-dev-practical-15". A secondary navigation bar contains "Overview", "Durability", "Settings" (which is underlined), and "Dataclips". The main content area is titled "ADMINISTRATION" and contains three sections: "Database Credentials" with a "View Credentials..." button, "Reset Database" with a "Reset Database..." button, and "Destroy Database" with a "Destroy Database..." button. Each section includes a brief description of the action.

postgresql-sinuous-86922 | Hi x +

data.heroku.com/datastores/1380143b-544b-4b52-afbd-cf616ab171f1#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...

# Postgres on Heroku

## 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://fivgqmxqetjwr: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 a Django Application

# Deploy a Django Application

We need to make some adjustments to our Django app before it can be run on Heroku

- 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 a Django Application - settings

- Some settings will differ between development and production versions of our app
- Environment-specific configurations should be stored in environment variables & not in the application's source code, i.e., `SECRET_KEY` in `settings.py`
- 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>`
- Install the `dotenv` Python module by running the command `pipenv install dotenv`
- To use environment variables locally:
  - In `settings.py`, declare the following:
    - `from dotenv import load_dotenv`
    - `load_dotenv()`
    - `SECRET_KEY = os.environ.get('SECRET_KEY')`



# Deploy a Django Application - settings

- Navigate to you Heroku app's settings page
- Heroku sets environments variables using config vars
- The `DATABASE_URL` config variable is automatically created when Heroku Postgres is added
- Set `SECRET_KEY` in config vars
- Resource: <https://devcenter.heroku.com/articles/config-vars>

## Config Vars

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

## Config Vars

Hide Config Vars

DATABASE\_URL

postgres://fivgqmxqetjwr:063fa9703ff:

✎ ✕

SECRET\_KEY

2vnv(eros&lywn-yz&yjc67ac04lv\$mf\*it1:

✎ ✕

KEY

VALUE

Add

# Deploy a Django Application - DB config

How do we connect to Postgres?

- Install the dj-database-url and psycopg2 Python modules
- 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) # put this line at the bottom of settings.py
```

- 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
- Don't forget to run your migrations: `heroku run python manage.py migrate`

# Deploy Django Application - static assets

- 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 - static assets

- 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 a 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