**Experiment No.: 08**

**Title: Working with Cloud services (Self-Learning)**

**Batch: IT A4 Roll no: 1914078 Experiment No.: 8**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Aim:** Exploring any one cloud service on any cloud platform

**Resources needed:** AWS/ Azure/ Google App Engine/ MongoDB Atlas

**Pre-Requisite:** Account information AWS/ Azure/ Google App Engine/ MongoDB Atlas

**Procedure:**

1. Each student will explore any one service on any cloud platform.
2. Each student will perform the selected service (IaaS/ PaaS/ SaaS) with documentation in the write-up format.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

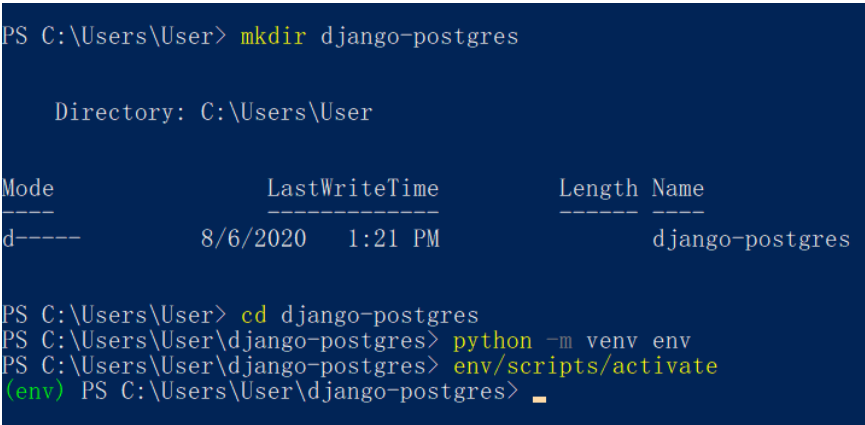
**RESULTS:**

**Project Aim:**

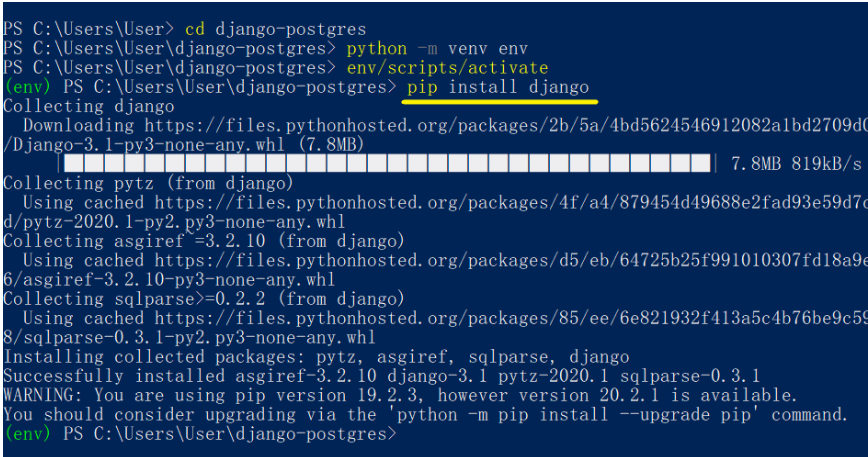
To build a file sharing application in React, Express using **MongoDB Atlas as the DBaaS** and deploy the application on Heroku (backend) and Netlify (frontend).

**STEPS:**

**Step 1:** Create a project directory, in this case I use “django-postgres”

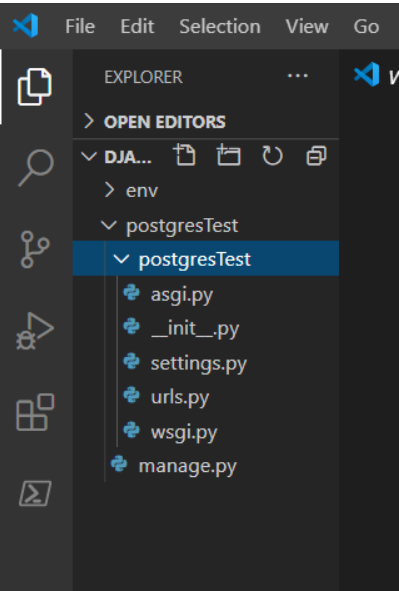


# Step 2: Install Django

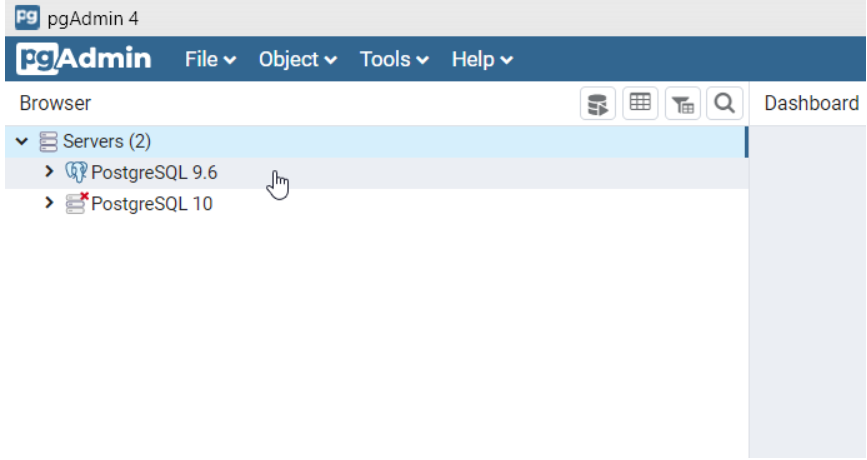


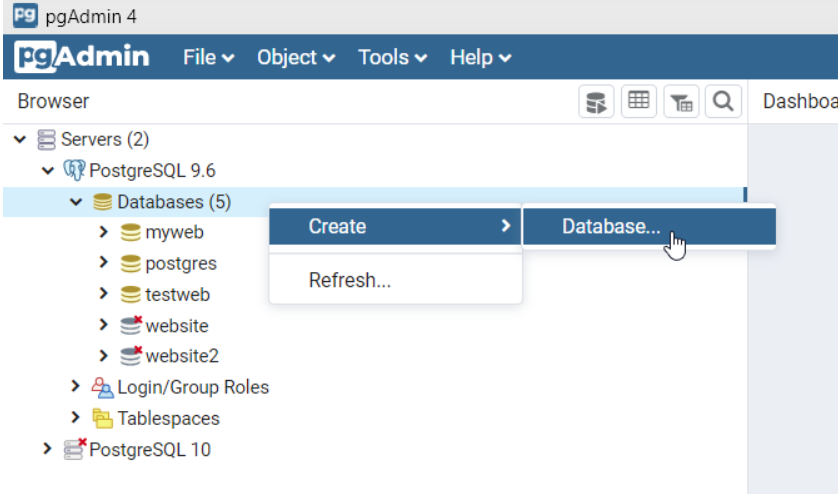
**Step3: Start Django Project**

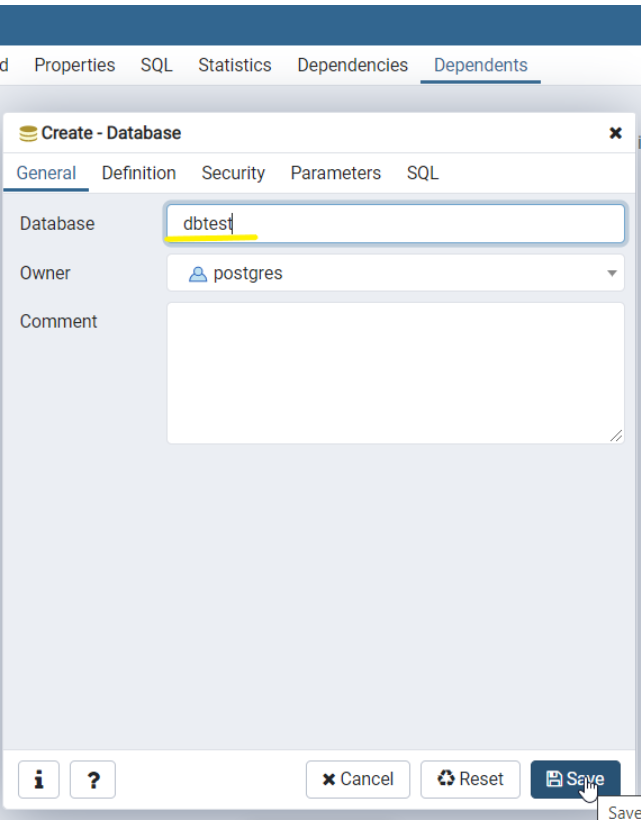


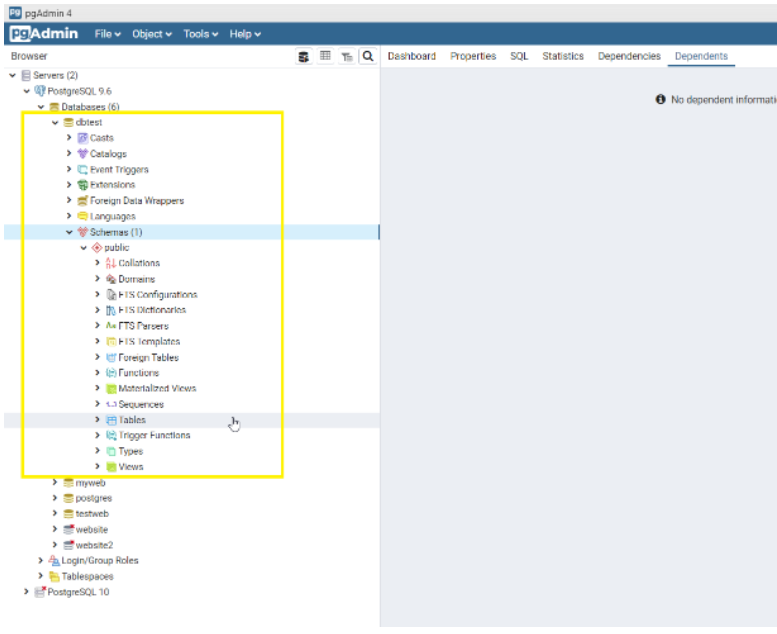


**Step 4: Setting up Database Server**

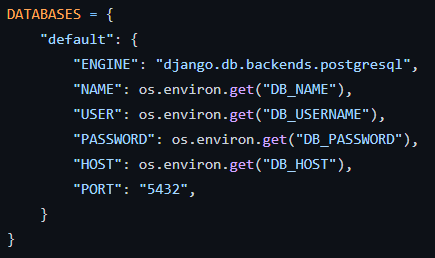








**Step 4: Connect to database from Django**

****

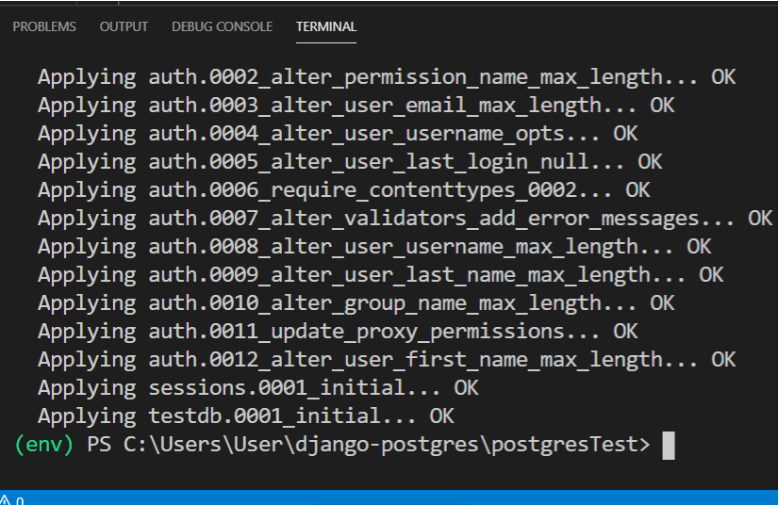
**Step 5: Migrate table to Postgresql database**

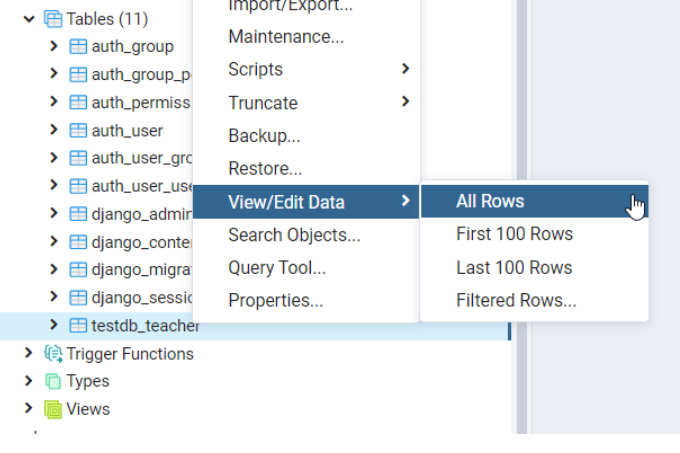
Now, we have successfully writen our table but the table isn’t being sent to PostgreSQL yet. So what we must do are the following procedures

makemigrations → To update and see the history or transaction happened in our table (We have to run this command everytime when something changes in models.py e.g. adding new table, change a field name, etc

migrate → The last step to submit or sent out our table into the database

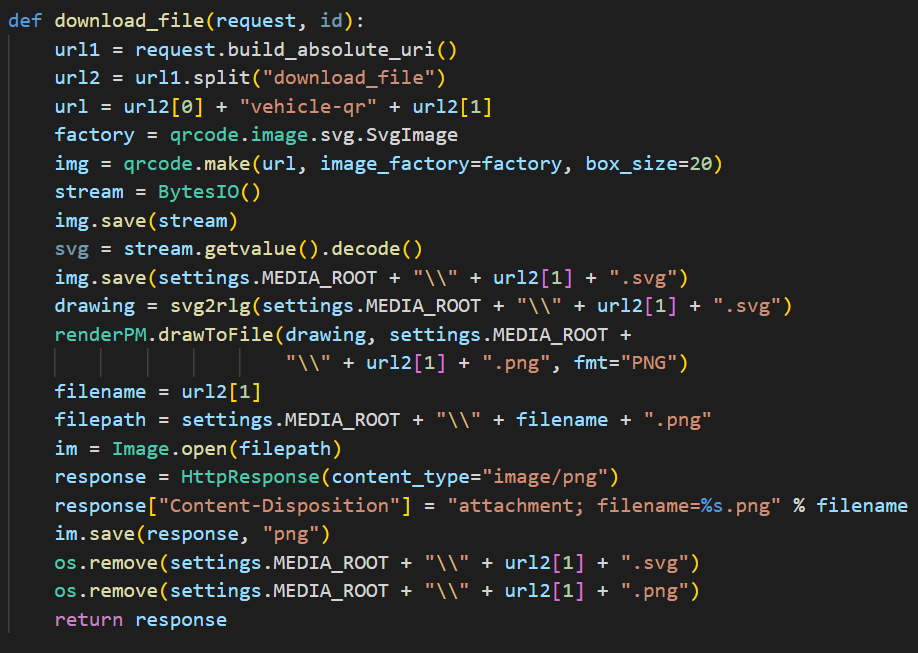




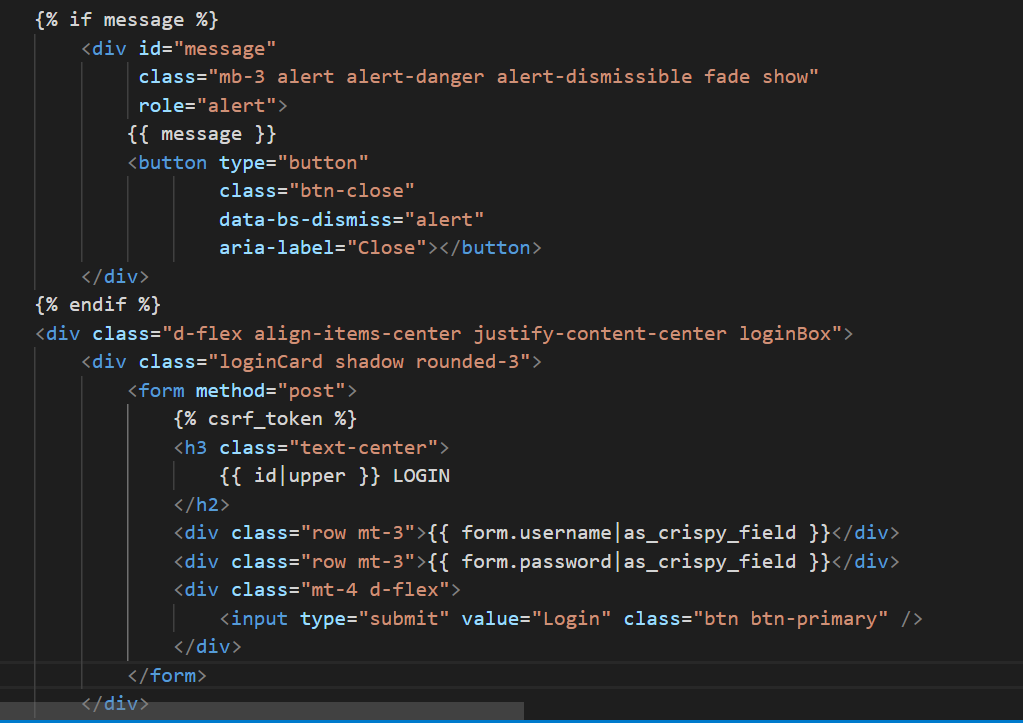


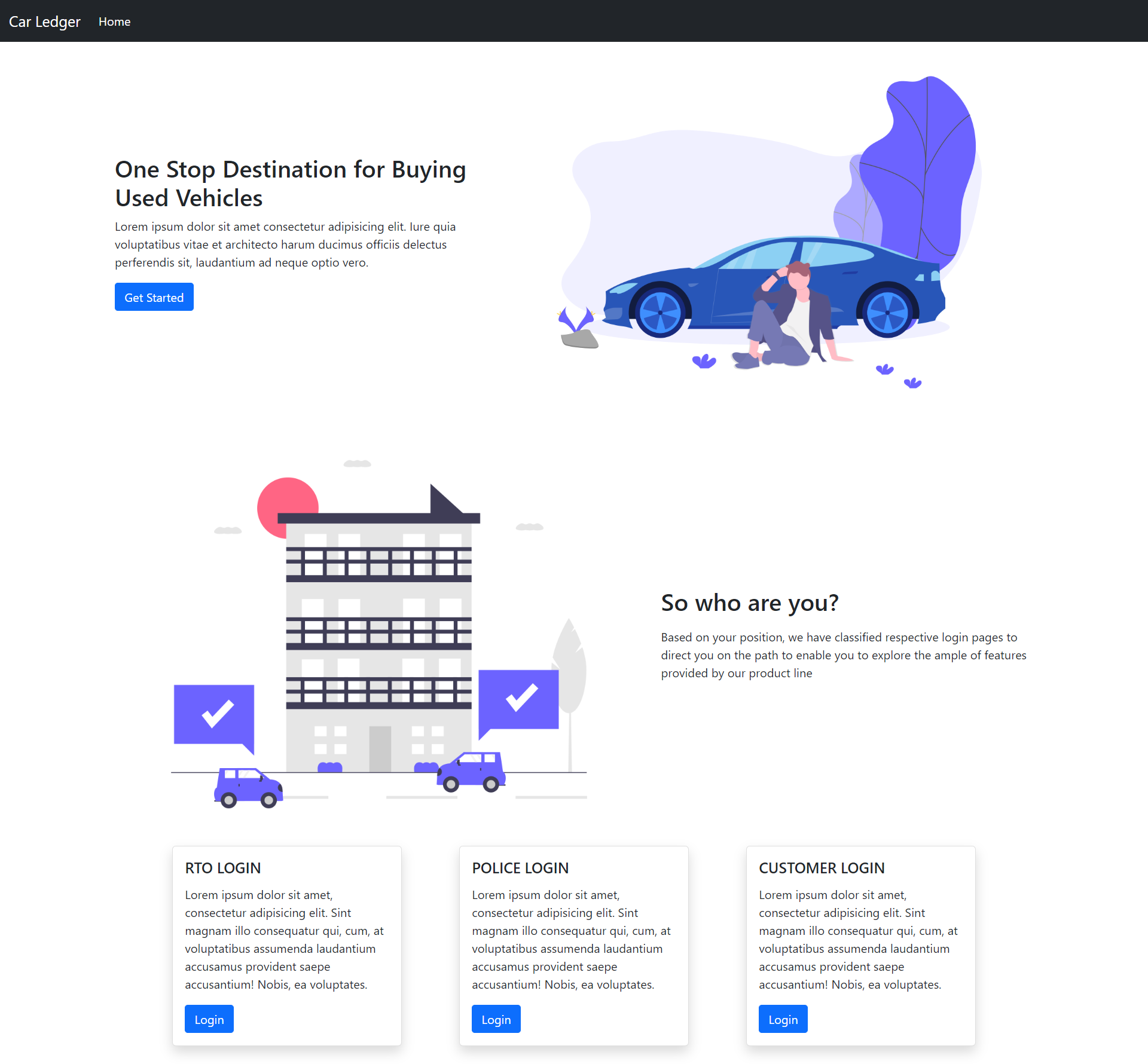
**Step 7: Build the frontend on Django**

Build the frontend app by integrating the deployed API endpoints as shown below.

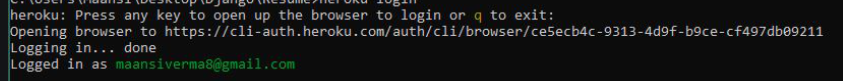


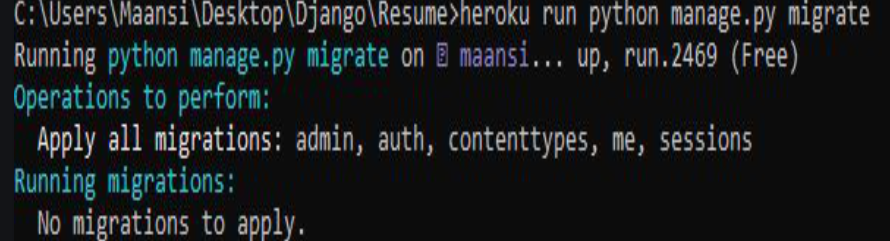






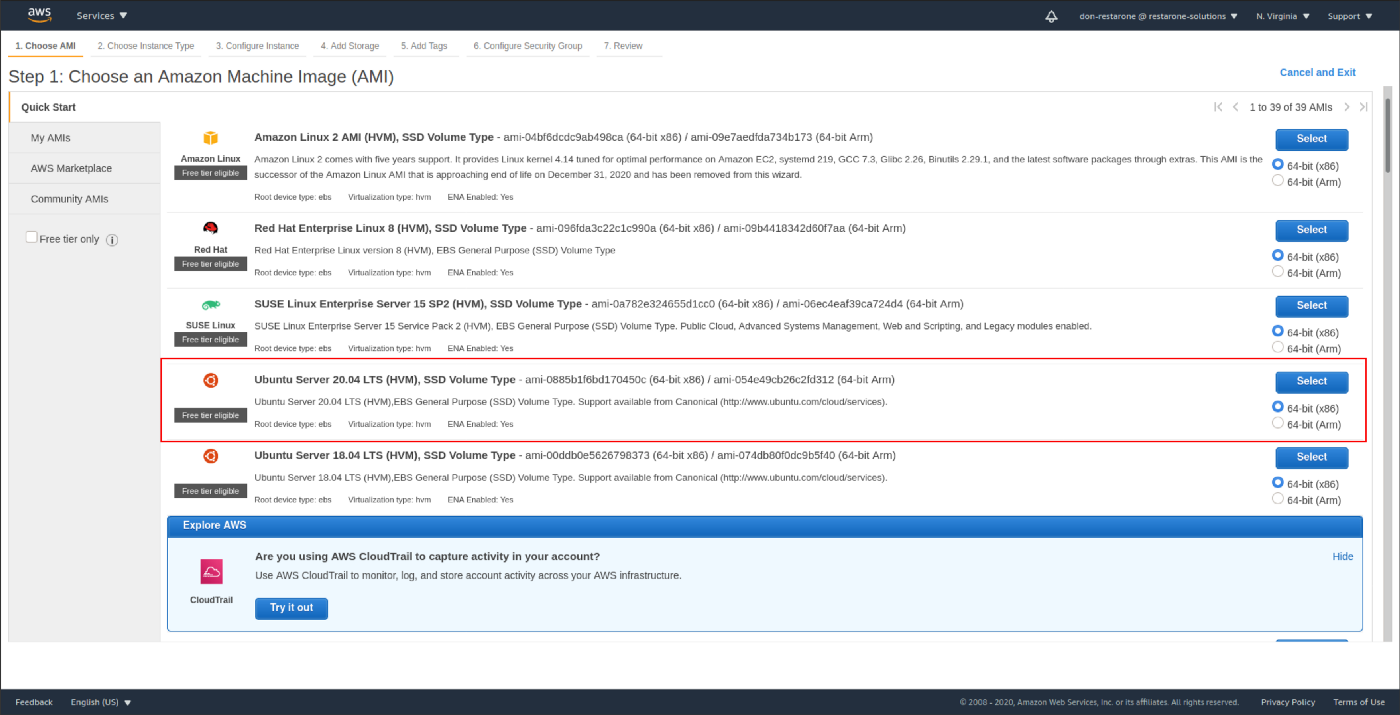
**Step 7: Deploy the frontend react app on Heroku**

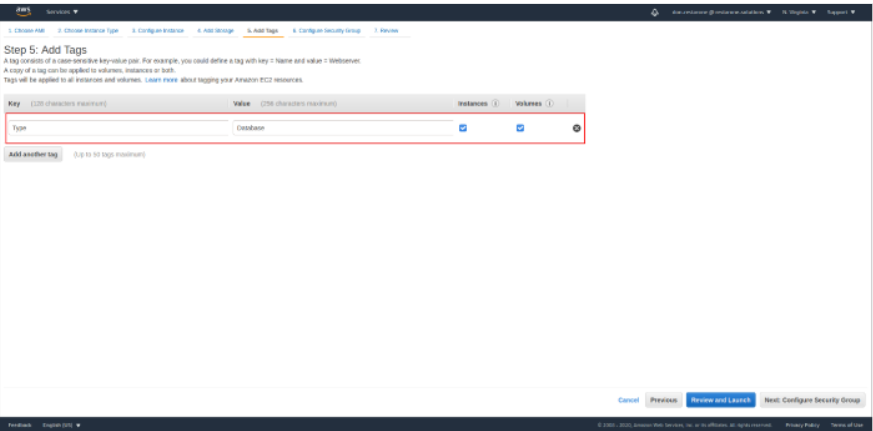




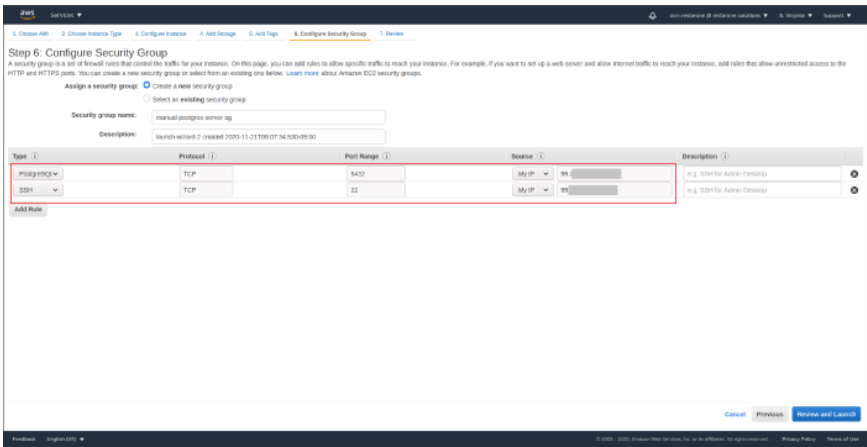
# Set Up the Infrastructure

Jump to the AWS EC2 console and provision a Linux server to host our DB. I’ll be choosing an Ubuntu 20.04 server because that is what I’m most accustomed to.

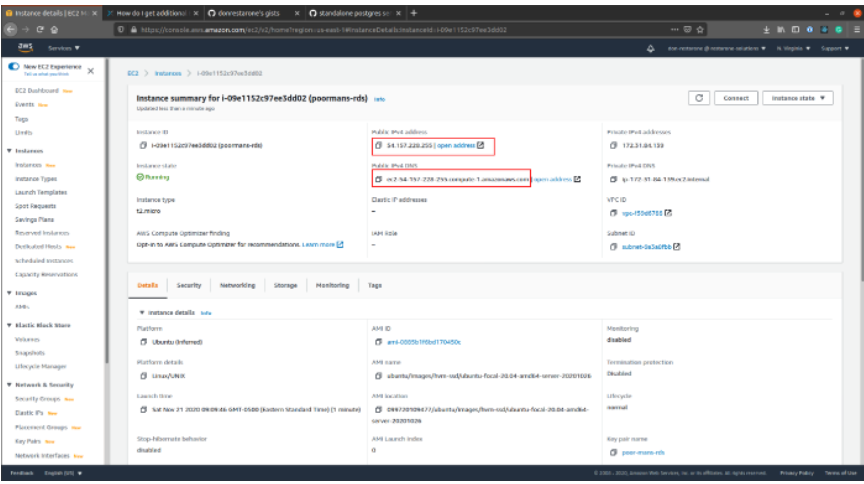


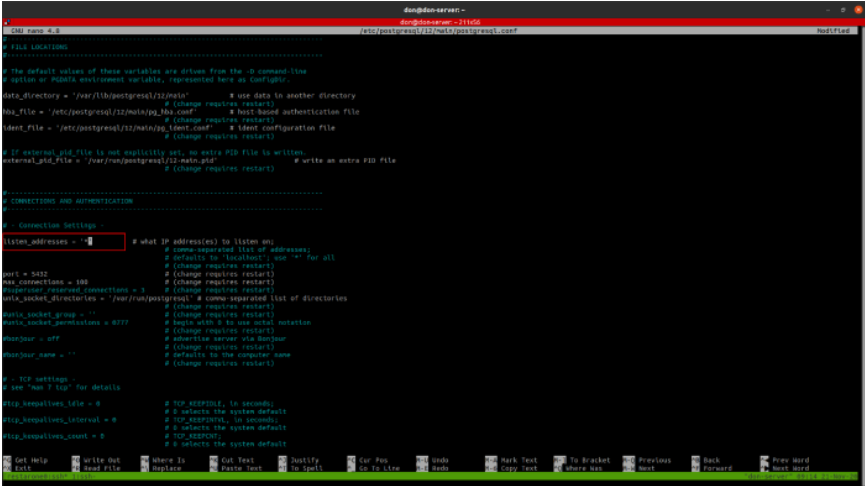


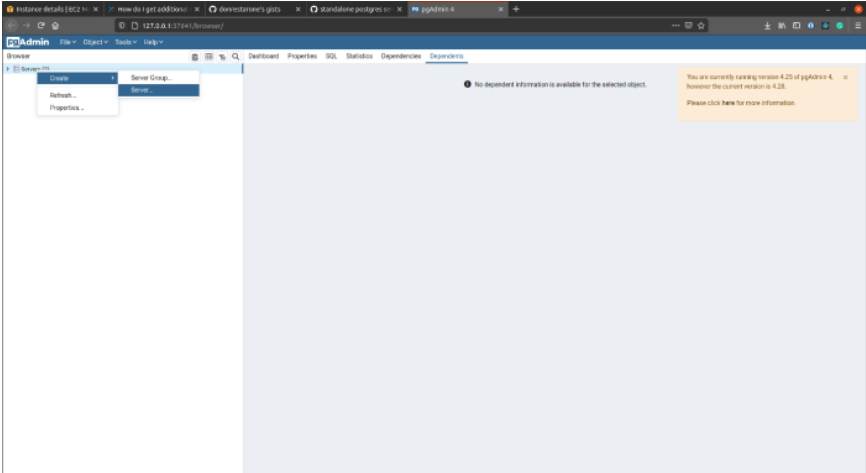
Next we will configure the security group to allow SSH and database access. I will open up the default Postgres port 5432 in anticipation of running the Postgres service there. For added security, I am limiting access to my public IP address. For production use, you should only allow database connections from the security group of your application server.



Once the instance has been launched, take note of the public IP address and public DNS name. We will be using the public IP address to SSH into the server for setting up Postgres. After that, we can use the DNS name to connect to the Postgres server.







**Outcomes: CO2:** Study the Evolution of Cloud Computing and its models

**Conclusion:**

We have successfully built a used cars tracking system based on Django integrated with Postgresql and AWS cloud with deployment on Heroku

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Grade: AA / AB / BB / BC / CC / CD /DD:** **Signature of faculty in-charge with Date**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**References:**

* https://car-ledger.herokuapp.com/