Here’s a stepbystep guide to deploy a 3tier application (frontend: Nginx, backend: Django, database: PostgreSQL) on an AWS EC2 instance running Ubuntu 24.04:

1. Launch an EC2 Instance

Go to the AWS Management Console.

Navigate to EC2 and click Launch Instance.

Choose Ubuntu Server 24.04 LTS (HVM), SSD Volume Type.

Select an appropriate instance type (e.g., t2.micro for free tier or a larger instance if needed).

Configure security groups to allow:

HTTP (port 80)

HTTPS (port 443, if needed)

SSH (port 22) – for accessing the instance

Launch the instance and download the private key (.pem file) for SSH access.

2. Connect to the EC2 Instance

Connect to the instance via SSH from your local terminal:

```

ssh i /path/to/yourkey.pem ubuntu@yourec2publicip

```

3. Update the System

Update and upgrade the package list to ensure everything is up to date:

```

sudo apt update

sudo apt upgrade y

```

4. Install Nginx (Frontend)

Install Nginx:

```

sudo apt install nginx y

```

Start and enable Nginx:

```

sudo systemctl start nginx

sudo systemctl enable nginx

```

Confirm Nginx is running by visiting the public IP of your instance (`http://yourec2publicip`) in a browser.

5. Install PostgreSQL (Database)

Install PostgreSQL:

```

sudo apt install postgresql postgresqlcontrib y

```

Set up the PostgreSQL database and user:

```

sudo u postgres psql

```

Inside the PostgreSQL shell, run:

```sql

CREATE DATABASE your\_db\_name;

CREATE USER your\_user WITH PASSWORD 'your\_password';

GRANT ALL PRIVILEGES ON DATABASE your\_db\_name TO your\_user;

\q

```

6. Install Python, Pip, and Django (Backend)

Install Python and pip:

```

sudo apt install python3 python3pip y

```

Install Django and PostgreSQL connector:

```

pip3 install django psycopg2binary

```

7. Clone Your Django Project

Navigate to your home directory or desired directory:

```

cd ~

```

Clone your Django project from GitHub:

```

git clone https://github.com/yourrepositorylink.git

cd yourrepositoryname

```

8. Configure Django to Use PostgreSQL

Edit the `settings.py` file in your Django project to configure PostgreSQL:

```

nano your\_project/settings.py

```

Replace the `DATABASES` section with your PostgreSQL configuration:

```python

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql',

'NAME': 'your\_db\_name',

'USER': 'your\_user',

'PASSWORD': 'your\_password',

'HOST': 'localhost',

'PORT': '5432',

}

}

```

9. Run Database Migrations

Apply the migrations to set up the database schema:

```

python3 manage.py migrate

```

10. Start Django Server for Testing

Run the Django development server to test your setup:

```

python3 manage.py runserver 0.0.0.0:8000

```

Visit `http://yourec2publicip:8000` to ensure the backend is running.

11. Configure Gunicorn (Application Server)

Install Gunicorn:

```

pip3 install gunicorn

```

Test Gunicorn by running it on your Django project:

```

gunicorn bind 0.0.0.0:8000 your\_project\_name.wsgi

```

12. Configure Nginx as a Reverse Proxy

Create an Nginx configuration file for your Django project:

```

sudo nano /etc/nginx/sitesavailable/your\_project

```

Add the following configuration:

```nginx

server {

listen 80;

server\_name your\_domain\_or\_ip;

location / {

proxy\_pass http://127.0.0.1:8000;

proxy\_set\_header Host $host;

proxy\_set\_header XRealIP $remote\_addr;

proxy\_set\_header XForwardedFor $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header XForwardedProto $scheme;

}

}

```

Save the file and enable the configuration:

```

sudo ln s /etc/nginx/sitesavailable/your\_project /etc/nginx/sitesenabled/

sudo nginx t # Test Nginx configuration

sudo systemctl restart nginx

```

13. Run Gunicorn as a Background Service

Create a Gunicorn systemd service file:

```

sudo nano /etc/systemd/system/gunicorn.service

```

Add the following content:

```ini

[Unit]

Description=gunicorn daemon

After=network.target

[Service]

User=ubuntu

Group=wwwdata

WorkingDirectory=/home/ubuntu/your\_project

ExecStart=/usr/local/bin/gunicorn workers 3 bind unix:/home/ubuntu/your\_project.sock your\_project\_name.wsgi:application

[Install]

WantedBy=multiuser.target

```

Start and enable Gunicorn:

```

sudo systemctl start gunicorn

sudo systemctl enable gunicorn

```

14. Open EC2 Security Group for Port 80

Go to your EC2 instance’s Security Group settings and make sure that HTTP (port 80) is open to allow traffic.

15. Test Your Application

Visit your EC2 instance’s public IP in a browser (`http://yourec2publicip`) to ensure that your Django application is served by Nginx.

16. (Optional) Set Up HTTPS

If you want to use HTTPS, you can configure a free SSL certificate using Let’s Encrypt:

```

sudo apt install certbot python3certbotnginx y

sudo certbot nginx d your\_domain

```

That’s it! You’ve deployed a 3tier application on AWS with Nginx, Django, and PostgreSQL. Let me know if you encounter any issues or need further help!