3 – Tier Architecture Deployment

Step-by-step 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:
```
```

```
[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.

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!