Deploying Django application using Azure Virtual machine

1. Setting Up Azure Virtual Machines

1.1. Create VM for Django Application

Go to Virtual Machines > Create.
Configure:

1. Name: eg.django-vm
2. Image: Ubuntu Server (LTS version).
3. Size: Choose based on your workload.
4. Authentication: Use SSH or a strong password.
5. Deploy the VM.

1.2. Create VM for PostgreSQL

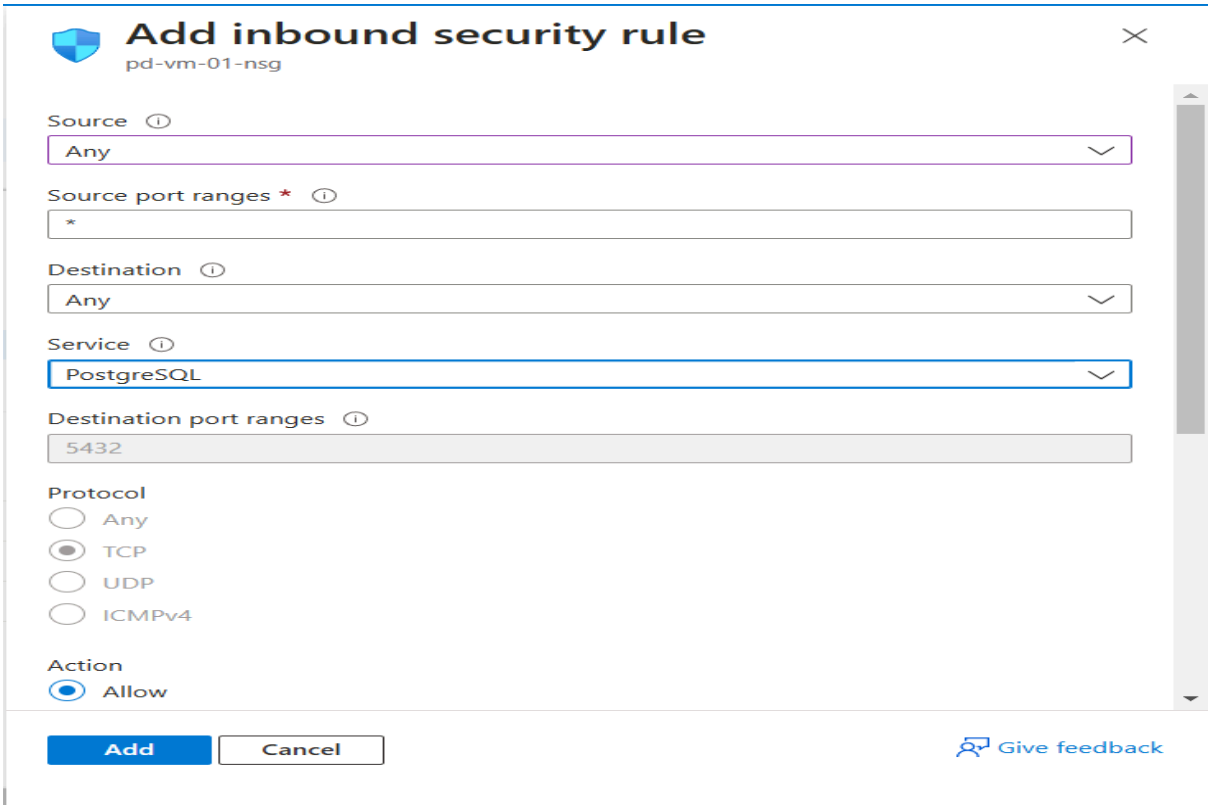
Go to Virtual Machines > Create.

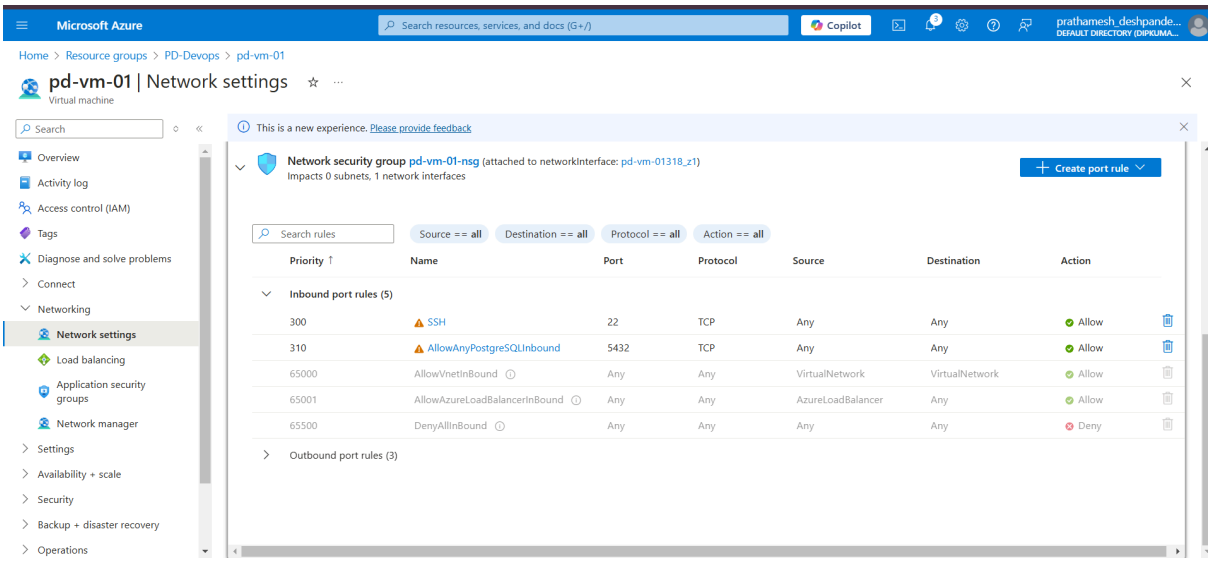
Configure:

1) Name: eg.postgresql-vm
2) Image: Ubuntu Server (LTS version).
3) Size: Choose based on your database workload.
4) Authentication: Use SSH or a strong password.
5) Deploy the VM.
6) Configure Azure Network Security Group:

   Modify the Network Security Group (NSG):

   a) In the Azure Portal, navigate to the NSG associated with your VM.
   b) Add an inbound security rule to allow TCP traffic on port 5432 from your backend VM's IP address or specific IPs.





2. Setting Up the PostgreSQL VM

2.1. Install PostgreSQL

1. **SSH into the PostgreSQL VM:**
   ssh azureuser@<postgresql-vm-public-ip>

```
C:\Windows\System32>ssh root1@40.81.241.80
root1@40.81.241.80's password:
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1018-azure x86_64)
```

2. **Update and install PostgreSQL:**

   sudo apt update && sudo apt upgrade -y

   sudo apt install postgresql postgresql-contrib -y

2.2. Configure PostgreSQL

1) **Log in as the PostgreSQL user:**

   sudo -i -u postgres psql

```
root1@pd-vm-01:~$ sudo -i -u postgres
postgres@pd-vm-01:~$
```

2) **Create a database and user:**

   CREATE DATABASE mydatabase;

```
postgres=# CREATE DATABASE fundoo_db;
CREATE DATABASE
```

   CREATE USER mydbuser WITH PASSWORD 'mypassword';

```
postgres=# CREATE USER Prathamesh  WITH PASSWORD 'root@123';
CREATE ROLE
```

   GRANT ALL PRIVILEGES ON DATABASE mydatabase TO mydbuser;

```
postgres=# GRANT ALL PRIVILEGES ON DATABASE fundoo_db TO Prathamesh;
GRANT
postgres=# \l
                                                  List of databases
   Name    |  Owner   | Encoding | Locale Provider | Collate |  Ctype  | ICU Locale | ICU Rules |    Access privileges
-----------+----------+----------+-----------------+---------+---------+------------+-----------+-------------------------
 fundoo_db | postgres | UTF8     | libc            | C.UTF-8 | C.UTF-8 |            |           | =Tc/postgres           +
           |          |          |                 |         |         |            |           | postgres=CTc/postgres  +
           |          |          |                 |         |         |            |           | prathamesh=CTc/postgres
 postgres  | postgres | UTF8     | libc            | C.UTF-8 | C.UTF-8 |            |           |
 template0 | postgres | UTF8     | libc            | C.UTF-8 | C.UTF-8 |            |           | =c/postgres            +
           |          |          |                 |         |         |            |           | postgres=CTc/postgres
 template1 | postgres | UTF8     | libc            | C.UTF-8 | C.UTF-8 |            |           | =c/postgres            +
           |          |          |                 |         |         |            |           | postgres=CTc/postgres
(4 rows)
```

```
postgres=# ALTER DATABASE fundoo_db OWNER TO Prathamesh;
ALTER DATABASE
postgres=# \l
                                                  List of databases
   Name    |   Owner    | Encoding | Locale Provider | Collate |  Ctype  | ICU Locale | ICU Rules |    Access privileges
-----------+------------+----------+-----------------+---------+---------+------------+-----------+-------------------------
 fundoo_db | prathamesh | UTF8     | libc            | C.UTF-8 | C.UTF-8 |            |           | =Tc/prathamesh         +
           |            |          |                 |         |         |            |           | prathamesh=CTc/prathamesh
 postgres  | postgres   | UTF8     | libc            | C.UTF-8 | C.UTF-8 |            |           |
 template0 | postgres   | UTF8     | libc            | C.UTF-8 | C.UTF-8 |            |           | =c/postgres            +
           |            |          |                 |         |         |            |           | postgres=CTc/postgres
 template1 | postgres   | UTF8     | libc            | C.UTF-8 | C.UTF-8 |            |           | =c/postgres            +
           |            |          |                 |         |         |            |           | postgres=CTc/postgres
(4 rows)
```

3) **Exit:**
   /q

2.3. Enable Internal Access

1) **Edit the PostgreSQL configuration:**

Update:

```
listen_addresses = '*'
```

2) **Allow connections from the Django VM:**

sudo nano /etc/postgresql/<version>/main/pg_hba.conf

Add:

```
host all  all   0.0.0.0/0  md5
```

3) **Enable PostgreSQL to Start on Boot**

sudo systemctl enable postgresql

4) **Restart PostgreSQL:**

```
sudo systemctl restart postgresql
```

3. Setting Up the Django VM

3.1. Install Required Software

1) **SSH into the Django VM:**
   ssh azureuser@<django-vm-public-ip>

2) **Install Python, pip, virtualenv**
   sudo apt update && sudo apt upgrade -y
   sudo apt install python3 python3-pip python3-venv

3) **Install PostgreSQL Development Libraries**
   sudo apt install libpq-dev -y

4) **Install PostgreSQL development headers and libraries**

(necessary for connecting Django to PostgreSQL)

sudo apt install libpq-dev -y

3.2. Clone Django Application

1. **Set up a virtual environment**
   python3 -m venv myenv
   source myenv/bin/activate

2. **Install Django and Gunicorn**

   pip install django gunicorn

3. **Clone your repository:**
   git clone <your-repo-url>
   cd <project-directory>

```
(myenv) Prathamesh@pd-backend:~$ git clone -b dev https://github.com/antimaYAD/FUNDOO-NOTES
Cloning into 'FUNDOO-NOTES'...
remote: Enumerating objects: 240, done.
remote: Counting objects: 100% (240/240), done.
remote: Compressing objects: 100% (180/180), done.
remote: Total 240 (delta 147), reused 126 (delta 58), pack-reused 0 (from 0)
Receiving objects: 100% (240/240), 67.90 KiB | 1.13 MiB/s, done.
Resolving deltas: 100% (147/147), done.
```

4. **Install dependencies:**
   pip install -r requirements.txt

```
(myenv) Prathamesh@pd-backend:~$ cd FUNDOO-NOTES/
(myenv) Prathamesh@pd-backend:~/FUNDOO-NOTES$ ls
README.md  fundoonote  label  manage.py  notes  pytest.ini  requirements.txt  user
(myenv) Prathamesh@pd-backend:~/FUNDOO-NOTES$ pip install -r requirements.txt
Collecting amqp==5.2.0 (from -r requirements.txt (line 1))
  Downloading amqp-5.2.0-py3-none-any.whl.metadata (8.9 kB)
Collecting anyio==3.6.2 (from -r requirements.txt (line 2))
  Downloading anyio-3.6.2-py3-none-any.whl.metadata (4.7 kB)
Collecting arrow==1.2.3 (from -r requirements.txt (line 5))
```

3.3. Configure Django for PostgreSQL

1) **Update the DATABASES setting in settings.py:**

```
DATABASES = { 'default': {
'ENGINE': 'django.db.backends.postgresql',
'NAME': 'mydatabase',
'USER': 'mydbuser',
'PASSWORD': 'mypassword',
'HOST': '10.0.2.4',
IP 'PORT': '5432', } }
```

```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'fundoo_db',
        'USER': 'prathamesh',
        'PASSWORD': 'root@123',
        'HOST': '40.81.241.80',  # PostgreSQL VM_instance's IP
        'PORT': '5432',  # Default PostgreSQL port
    }
}
```

2) Install Postgresql Client

sudo apt install postgresql-client

```
(myenv) Prathamesh@pd-backend:~/FUNDOO-NOTES/fundoonote$ sudo apt install postgresql-client
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  postgresql-client-16 postgresql-client-common
Suggested packages:
  postgresql-16 postgresql-doc-16
The following NEW packages will be installed:
  postgresql-client postgresql-client-16 postgresql-client-common
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 1319 kB of archives.
After this operation, 4235 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 postgresql-client-common all 257build1.1 [36.4 kB]
Get:2 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 postgresql-client-16 amd64 16.4-0ubuntu0.24.04.2 [1271 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 postgresql-client all 16+257build1.1 [11.6 kB]
Fetched 1319 kB in 1s (1220 kB/s)
```

3) Test the Connection with Database

Test the database connection with the following command

psql -U <db-user> -d fundoo_db -h <postgre-vm-ip>

4) Apply migrations :

python manage.py migrate

```
(myenv) Prathamesh@pd-backend:~/FUNDOO-NOTES$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, django_celery_beat, django_celery_results, label, notes, sessions, user
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0001_initial... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying user.0001_initial... OK
```

3.4. Run the application Locally

python manage.py runserver 0.0.0.0:8000

3.5. Configure the daemon service file

1) **Create a Service File:**
   a) **Use a text editor to create the Gunicorn service file:**

The service files are usually located in /etc/systemd/system/. You'll create your custom service file there.

sudo nano /etc/systemd/system/<name>.service

   b) **Add the following content, replacing placeholders with actual paths and values:**

Description: A short description of your service.

After: Defines when the service should start, such as after the network is up.

User: The user that will run the service (typically your system user).

Group: The group for file permissions.

WorkingDirectory: The location where your project files reside.

ExecStart: The command to start your application (in this case, Gunicorn).

Restart=always: Automatically restarts the service if it crashes.

Environment: Use to define environment variables like Django settings.

```
 GNU nano 7.2                                  django_app.service *
[Unit]
Description=Fundoo Notes Django Application
After=network.target

[Service]
User=Prathamesh
Group=Prathamesh
WorkingDirectory=/home/Prathamesh/fundoo-notes/fundoo-notes
ExecStart=/home/Prathamesh/myenv/bin/python /home/Prathamesh/fundoo-notes/fundoo-notes/manage.py runserver 0.0.0.0:8000
Restart=always
Environment=PYTHONUNBUFFERED=1

[Install]
WantedBy=multi-user.target
```

2) **Enable and Start the Service**
   a) **Reload the systemd daemon to recognize the new service:**
   sudo systemctl daemon-reload

b) **Start the Gunicorn service:**
sudo systemctl start fundoo-service

c) **Enable the service to start on boot:**
sudo systemctl enable gunicorn

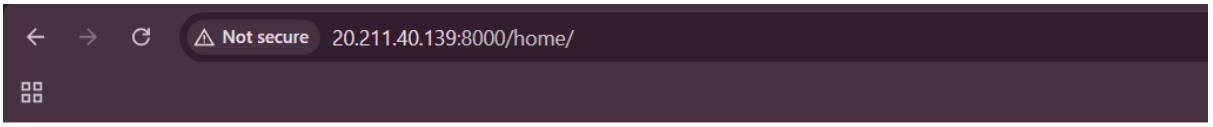
d) **Verify that the service is running:**

sudo systemctl status gunicorn

The output should show Active:active (running).

4. Verify Deployment

Once the setup is complete, verify that your Django application is running correctly by accessing it via its public IP address or domain name.



Welcome, to Fundoo notes Prathamesh Deshpande!

- **Perform API testing**

We can perform api testing using swagger to confirm our applications is running perfectly