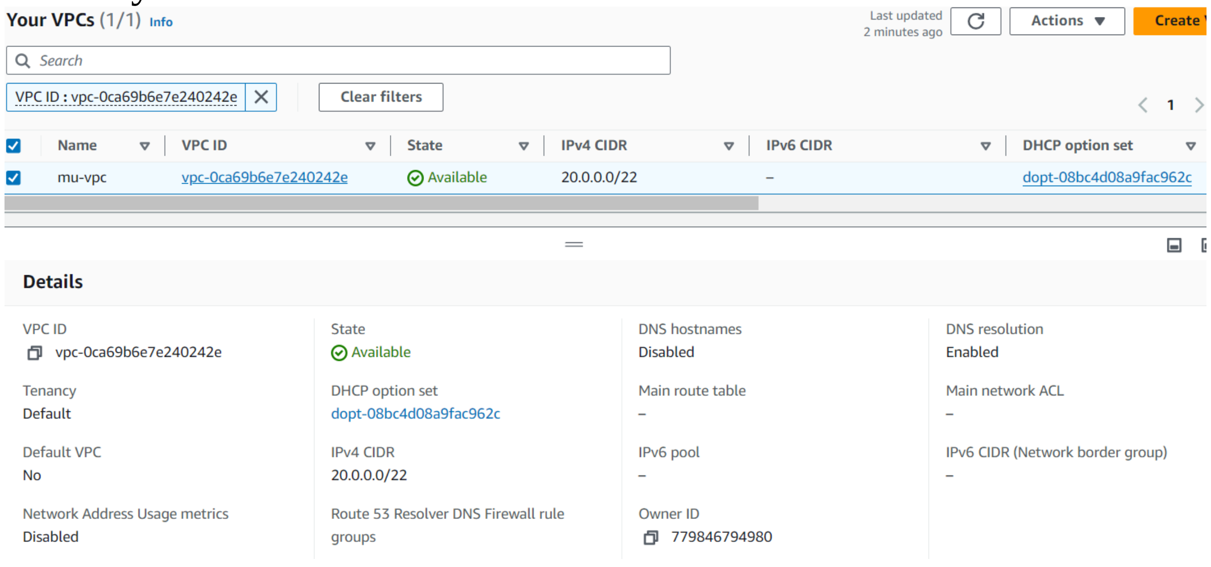
3tier application deployment on cloud

You need to deploy a 3tier application on cloud.
For this, first you need to create the cloud infrastructure as follows: Your own Virtual Private Cloud(Figure out how to choose an appropriate IP range as per your usecase), 2 public subnets, 2 private subnets and corresponding route tables, configure Internet Gateway, NAT gateway, 3 EC2 Instances, 1 Frontend in public, 2 Backend in private, 3 Database in private.

1. Create the VPC

Step: Go to the AWS Management Console, navigate to "VPC,"and click "Create VPC."
IP Range: Choose an appropriate CIDR block. For example, `20.0.0.0/22` allows for 1022 IP addresses, sufficient for future scaling.
VPC Name: Assign a name like `MyVPC`.
Tenancy: Default.



Your VPCs (1/1) Info

VPC ID : vpc-0ca69b6e7e240242e

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set
mu-vpc	vpc-0ca69b6e7e240242e	Available	20.0.0.0/22	–	dopt-08bc4d08a9fac962c

Details

VPC ID vpc-0ca69b6e7e240242e	State Available	DNS hostnames Disabled	DNS resolution Enabled
Tenancy Default	DHCP option set dopt-08bc4d08a9fac962c	Main route table –	Main network ACL –
Default VPC No	IPv4 CIDR 20.0.0.0/22	IPv6 pool –	IPv6 CIDR (Network border group) –
Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups	Owner ID 779846794980	

2. Create Subnets

Step: Create 2 public and 2 private subnets, each within a different Availability Zone (AZ) for high availability.

Public Subnet 1:

CIDR block: `20.0.0.0/24`

Availability Zone: Select `ap-southeast-1a` or any AZ in your region.

Enable Autoassign Public IPv4: Yes.

Public Subnet 2:

CIDR block: `20.0.1.0/24`

Availability Zone: `uap-southeast-1b` or another AZ.

Enable Autoassign Public IPv4: Yes.

Private Subnet 1:

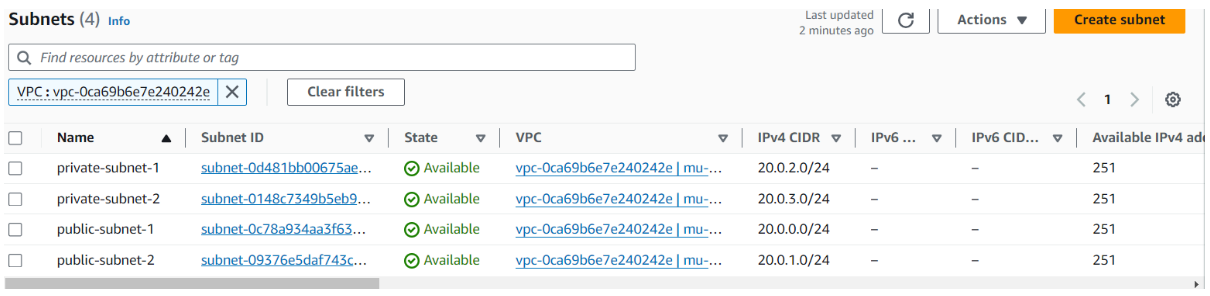
CIDR block: `20.0.2.0/24`

Availability Zone: `uap-southeast-1a`.

Private Subnet 2:

CIDR block: `20.0.3.0/24`

Availability Zone: `ap-southeast-1b`.


Subnets (4) Info

Last updated 2 minutes ago

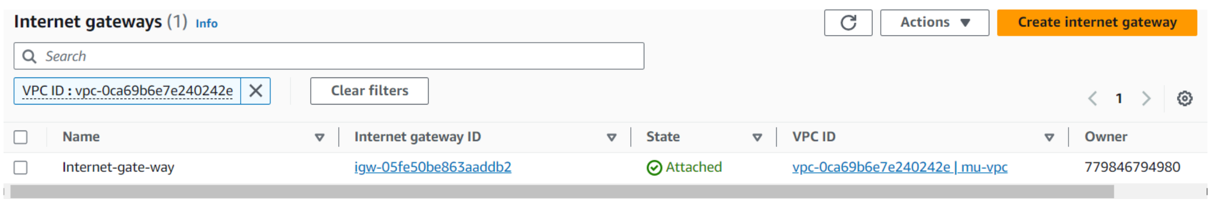
VPC : vpc-0ca69b6e7e240242e

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 ...	IPv6 CID...	Available IPv4 ad
private-subnet-1	subnet-0d481bb00675ae...	Available	vpc-0ca69b6e7e240242e | mu-...	20.0.2.0/24	–	–	251
private-subnet-2	subnet-0148c7349b5eb9...	Available	vpc-0ca69b6e7e240242e | mu-...	20.0.3.0/24	–	–	251
public-subnet-1	subnet-0c78a934aa3f63...	Available	vpc-0ca69b6e7e240242e | mu-...	20.0.0.0/24	–	–	251
public-subnet-2	subnet-09376e5daf743c...	Available	vpc-0ca69b6e7e240242e | mu-...	20.0.1.0/24	–	–	251

3. Create an Internet Gateway

Step: Go to "Internet Gateways" in the VPC dashboard and click "Create Internet Gateway."

Attach to VPC: After creating it, select the VPC created in Step 1 (`MyVPC`).


Internet gateways (1) Info

VPC ID : vpc-0ca69b6e7e240242e

Name	Internet gateway ID	State	VPC ID	Owner
Internet-gate-way	igw-05fe50be863aaddb2	Attached	vpc-0ca69b6e7e240242e | mu-vpc	779846794980

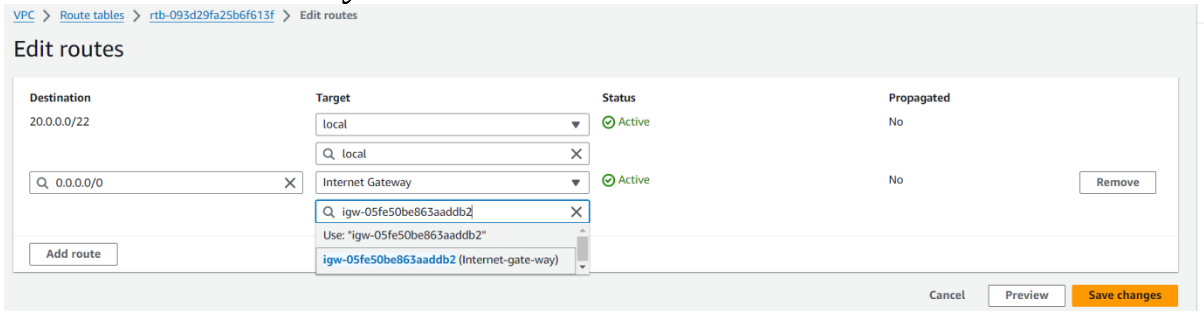
4. Create Route Tables

Public Route Table:

Step: Go to "Route Tables" and create a route table for public subnets.

Association: Associate this route table with both public subnets (`20.0.0.0/24` and `20.0.1.0/24`).

Route: Add a route that sends all traffic (`0.0.0.0/0`) to the Internet Gateway.



VPC > Route tables > rtb-093d29fa25b6f613f > Edit routes

Edit routes

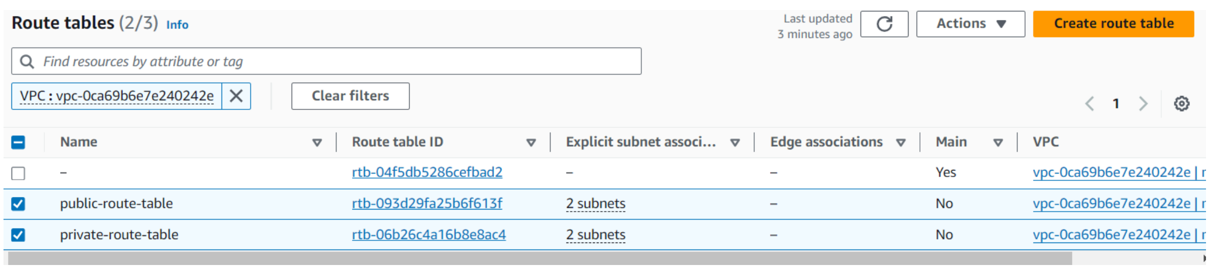
Destination	Target	Status	Propagated	
20.0.0.0/22	local	Active	No	
0.0.0.0/0	Internet Gateway: igw-05fe50be863aaddb2	Active	No	Remove

Use: "igw-05fe50be863aaddb2"
igw-05fe50be863aaddb2 (Internet-gate-way)

Private Route Table:

Step: Create a route table for private subnets.

Association: Associate this route table with both private subnets (`20.0.2.0/24` and `20.0.3.0/24`).



Route tables (2/3) Info

Last updated 3 minutes ago

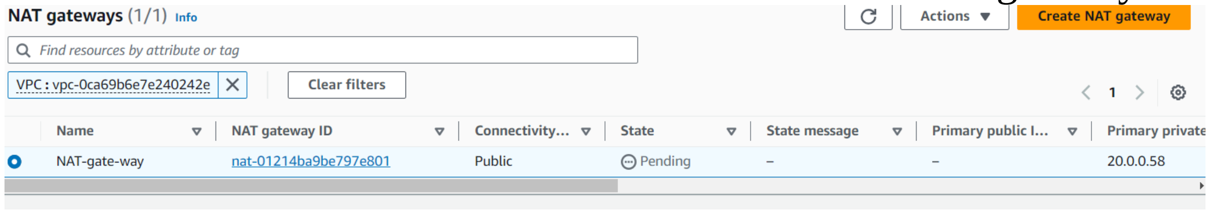
VPC : vpc-0ca69b6e7e240242e

	Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC
☐	–	rtb-04f5db5286cefbad2	–	–	Yes	vpc-0ca69b6e7e240242e | r
✓	public-route-table	rtb-093d29fa25b6f613f	2 subnets	–	No	vpc-0ca69b6e7e240242e | r
✓	private-route-table	rtb-06b26c4a16b8e8ac4	2 subnets	–	No	vpc-0ca69b6e7e240242e | r

5. Create a NAT Gateway

Step: Go to "NAT Gateways" and create a new NAT gateway in one of the public subnets (e.g., `10.0.1.0/24`).

Elastic IP: Allocate an Elastic IP and attach it to the NAT gateway.

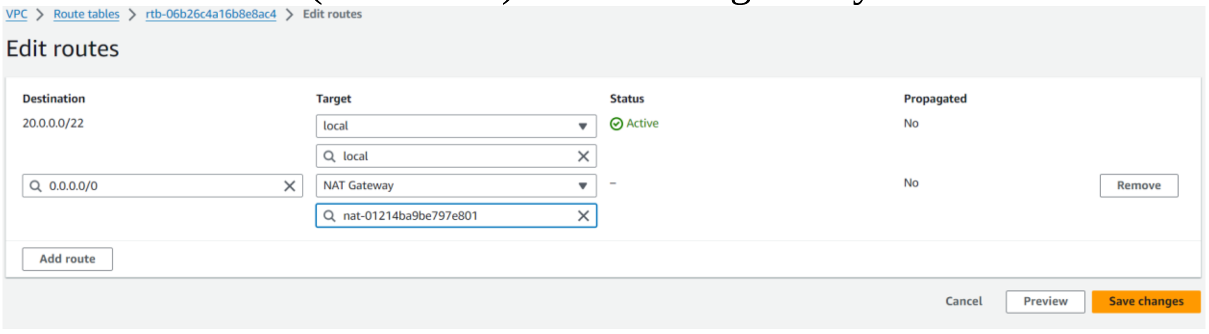


NAT gateways (1/1) Info

VPC : vpc-0ca69b6e7e240242e

	Name	NAT gateway ID	Connectivity...	State	State message	Primary public I...	Primary private
●	NAT-gate-way	nat-01214ba9be797e801	Public	Pending	–	–	20.0.0.58

Route for Private Subnets: In the private route table, add a route that sends all traffic (`0.0.0.0/0`) to the NAT gateway.



VPC > Route tables > rtb-06b26c4a16b8e8ac4 > Edit routes

Edit routes

Destination	Target	Status	Propagated	
20.0.0.0/22	local	Active	No	
0.0.0.0/0	NAT Gateway nat-01214ba9be797e801	–	No	Remove

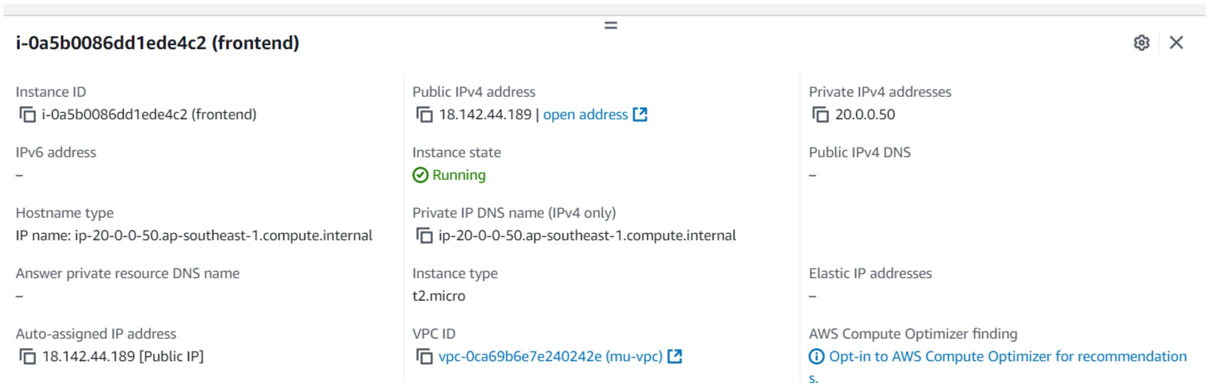
6. Launch EC2 Instances

Frontend EC2 (in Public Subnet):

AMI: Choose Ubuntu 24.04 or your preferred OS.

Subnet: Choose `10.0.1.0/24` (Public Subnet 1).

Security Group: Allow HTTP (port 80) and SSH (port 22).



i-0a5b0086dd1ede4c2 (frontend)

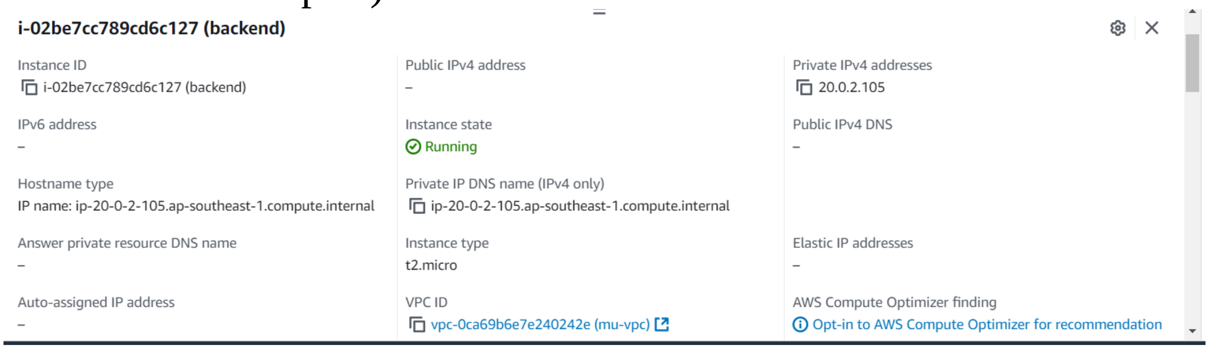
Instance ID: i-0a5b0086dd1ede4c2 (frontend)	Public IPv4 address: 18.142.44.189 | open address	Private IPv4 addresses: 20.0.0.50
IPv6 address: –	Instance state: Running	Public IPv4 DNS: –
Hostname type: IP name: ip-20-0-0-50.ap-southeast-1.compute.internal	Private IP DNS name (IPv4 only): ip-20-0-0-50.ap-southeast-1.compute.internal	
Answer private resource DNS name: –	Instance type: t2.micro	Elastic IP addresses: –
Auto-assigned IP address: 18.142.44.189 [Public IP]	VPC ID: vpc-0ca69b6e7e240242e (mu-vpc)	AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations.

Backend EC2 (in Private Subnets):

AMI: Choose Ubuntu 24.04 for backend instances.

Subnet: Choose `10.0.3.0/24` (Private Subnet 1) and `10.0.4.0/24` (Private Subnet 2).

Security Group: Allow traffic only from the frontend instance (via HTTP or custom port).

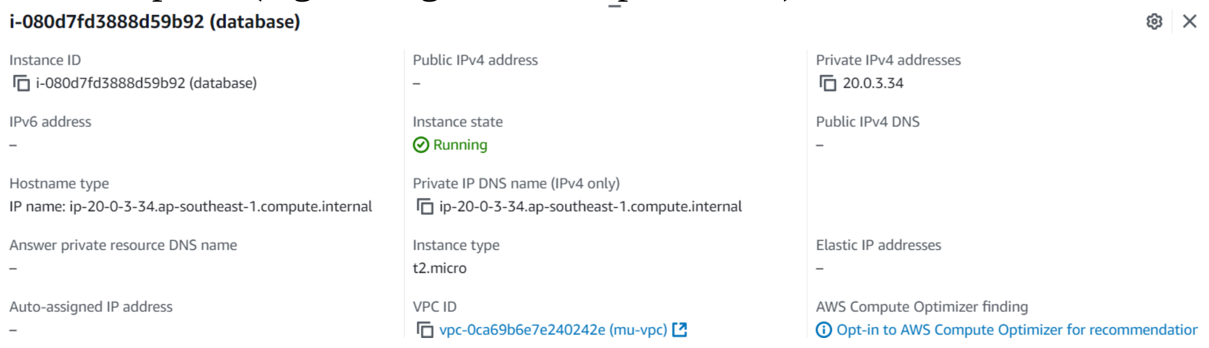


i-02be7cc789cd6c127 (backend)

Instance ID: i-02be7cc789cd6c127 (backend)	Public IPv4 address: –	Private IPv4 addresses: 20.0.2.105
IPv6 address: –	Instance state: Running	Public IPv4 DNS: –
Hostname type: IP name: ip-20-0-2-105.ap-southeast-1.compute.internal	Private IP DNS name (IPv4 only): ip-20-0-2-105.ap-southeast-1.compute.internal	
Answer private resource DNS name: –	Instance type: t2.micro	Elastic IP addresses: –
Auto-assigned IP address: –	VPC ID: vpc-0ca69b6e7e240242e (mu-vpc)	AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendation

Database EC2 (in Private Subnets):

AMI: Choose Ubuntu 24.04 or a preconfigured PostgreSQL image.
Subnet: Use private subnets (`10.0.3.0/24` and `10.0.4.0/24`).
Security Group: Allow only backend servers to access the database ports (e.g., PostgreSQL on port 5432).



i-080d7fd3888d59b92 (database)

Instance ID i-080d7fd3888d59b92 (database)	Public IPv4 address –	Private IPv4 addresses 20.0.3.34
IPv6 address –	Instance state Running	Public IPv4 DNS –
Hostname type IP name: ip-20-0-3-34.ap-southeast-1.compute.internal	Private IP DNS name (IPv4 only) ip-20-0-3-34.ap-southeast-1.compute.internal	
Answer private resource DNS name –	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address –	VPC ID vpc-0ca69b6e7e240242e (mu-vpc)	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendatior

7. Configure Security Groups

Frontend Security Group: Allow HTTP (80), HTTPS (443), and SSH (22) from your IP and the backend security group for internal traffic.

Backend Security Group: Allow HTTP or custom app port traffic only from the frontend.

Database Security Group: Restrict access to PostgreSQL (port 5432) from backend servers only.

8. Connect and Test

Step-1: Frontend: Connect to the public EC2 instance via SSH using its public IP.

Step-1.1: create a file add paste pem.key in that file
Step-1.2: change only read permission for user

Step-2: Backend: Connect from the frontend to the backend instances using private IPs.

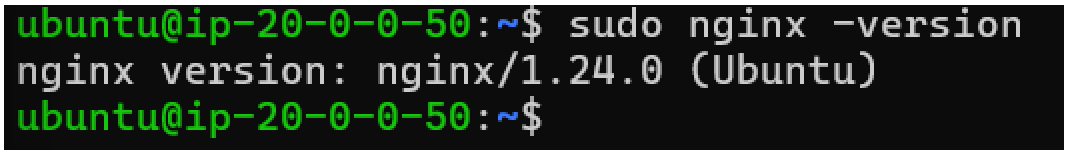
Step-2.1: create a file add paste pem.key in that file
Step-3.2: change only read permission for user

Step-3: Database: From the backend EC2, ensure you can connect to the database EC2 over its private IP.

Configure EC2 Instances

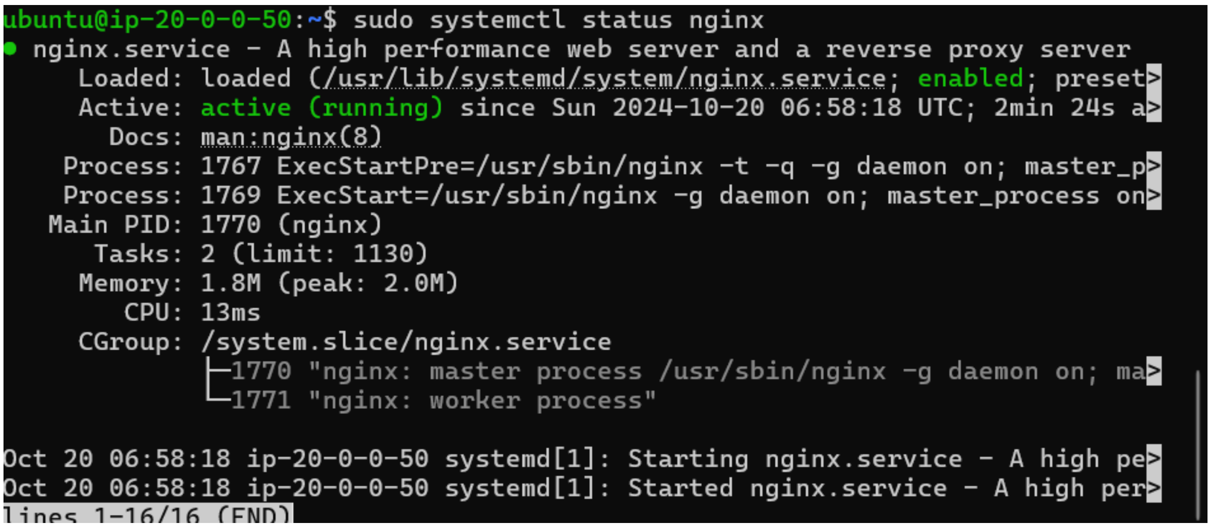
Frontend: Set up Nginx on the frontend EC2 instance.

Step-1 : sudo apt update
Step-2 : sudo apt install nginx -y
Step-3 : sudo nginx -version



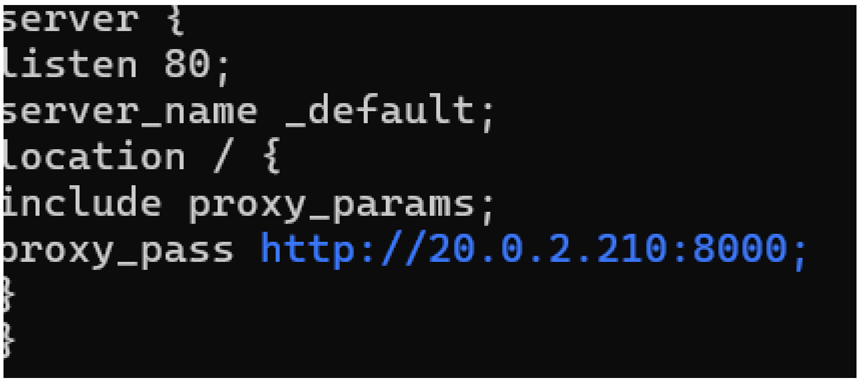
```
ubuntu@ip-20-0-0-50:~$ sudo nginx -version
nginx version: nginx/1.24.0 (Ubuntu)
ubuntu@ip-20-0-0-50:~$
```

To manage the nginx commands
Step-4: Sudo systemctl start nginx
        Sudo systemctl stop nginx
        Sudo systemctl enable nginx
        Sudo systemctl restart nginx
        Sudo systemctl status nginx



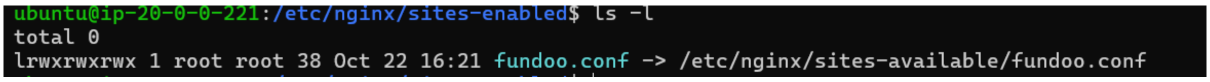
```
ubuntu@ip-20-0-0-50:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset>
     Active: active (running) since Sun 2024-10-20 06:58:18 UTC; 2min 24s a>
       Docs: man:nginx(8)
    Process: 1767 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_p>
    Process: 1769 ExecStart=/usr/sbin/nginx -g daemon on; master_process on>
   Main PID: 1770 (nginx)
      Tasks: 2 (limit: 1130)
     Memory: 1.8M (peak: 2.0M)
        CPU: 13ms
     CGroup: /system.slice/nginx.service
             ├─1770 "nginx: master process /usr/sbin/nginx -g daemon on; ma>
             └─1771 "nginx: worker process"

Oct 20 06:58:18 ip-20-0-0-50 systemd[1]: Starting nginx.service - A high pe>
Oct 20 06:58:18 ip-20-0-0-50 systemd[1]: Started nginx.service - A high per>
lines 1-16/16 (END)
```

Configure nginx
sudo nano /etc/nginx/sites-available/fundoo-conf



```
server {
listen 80;
server_name _default;
location / {
include proxy_params;
proxy_pass http://20.0.2.210:8000;
}
}
```

Unlink default
sudo ln -s /etc/nginx/sites-available/fundoo-conf /etc/nginx/sites-enabled/



```
ubuntu@ip-20-0-0-221:/etc/nginx/sites-enabled$ ls -l
total 0
lrwxrwxrwx 1 root root 38 Oct 22 16:21 fundoo.conf -> /etc/nginx/sites-available/fundoo.conf
```

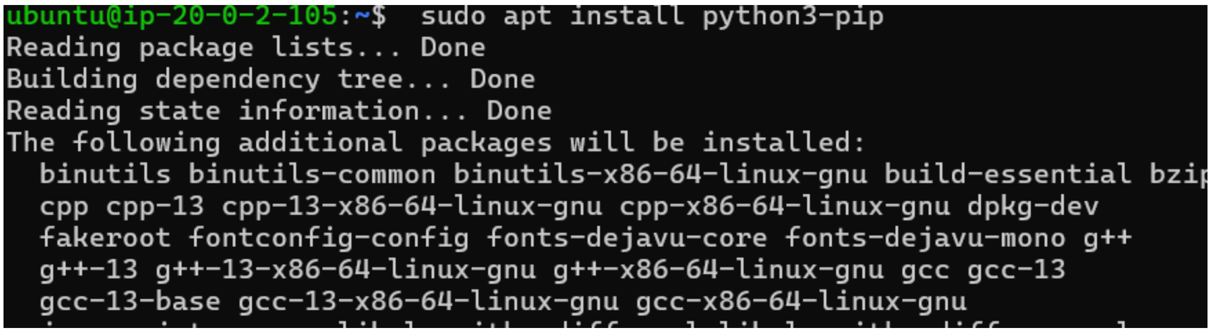
Restart Nginx:
sudo systemctl restart nginx

Reference-link:https://docs.vultr.com/how-to-install-nginx-web-server-on-ubuntu-24-04

Backend: Install Django and configure it.

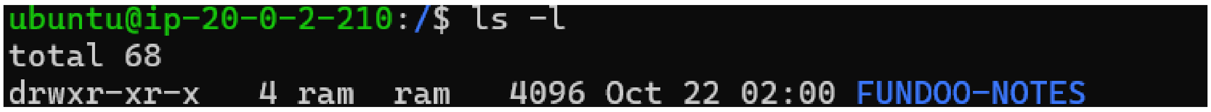
Step:1:$ sudo apt update && sudo apt upgrade

Step:2:$ sudo apt install python3-pip



```
ubuntu@ip-20-0-2-105:~$  sudo apt install python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential bzip
  cpp cpp-13 cpp-13-x86-64-linux-gnu cpp-x86-64-linux-gnu dpkg-dev
  fakeroot fontconfig-config fonts-dejavu-core fonts-dejavu-mono g++
  g++-13 g++-13-x86-64-linux-gnu g++-x86-64-linux-gnu gcc gcc-13
  gcc-13-base gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu
```

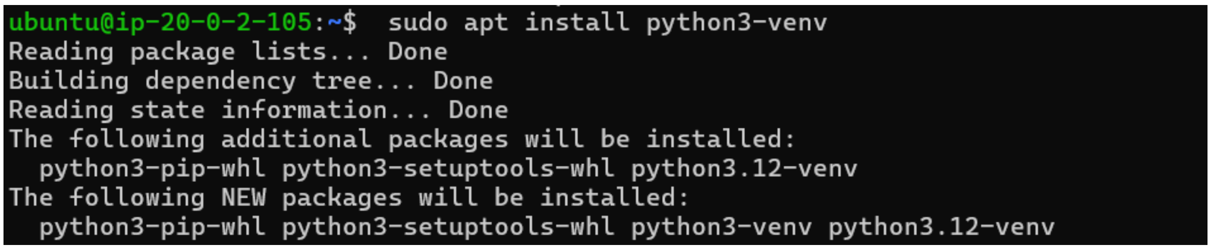
Sudo git clone -b dev https://github.com/Aniket26559/Aws_test.git


```
ubuntu@ip-20-0-2-210:/$ ls -l
total 68
drwxr-xr-x   4 ram  ram   4096 Oct 22 02:00 FUNDOO-NOTES
```

Using Python Virtual Environment

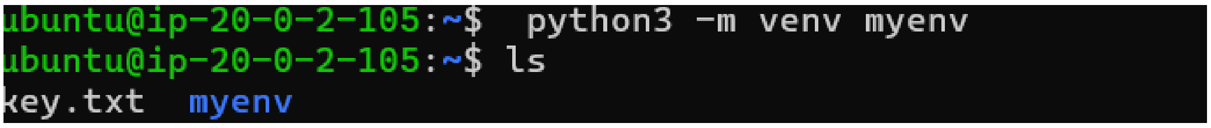
Step 1: Installing Python3 Virtual Environment

$ sudo apt install python3-venv


```
ubuntu@ip-20-0-2-105:~$  sudo apt install python3-venv
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-pip-whl python3-setuptools-whl python3.12-venv
The following NEW packages will be installed:
  python3-pip-whl python3-setuptools-whl python3-venv python3.12-venv
```

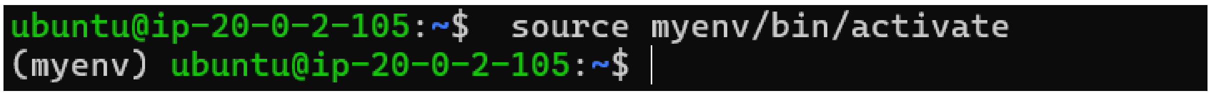
Step 2: Create a Virtual Environment

$ python3 -m venv myenv

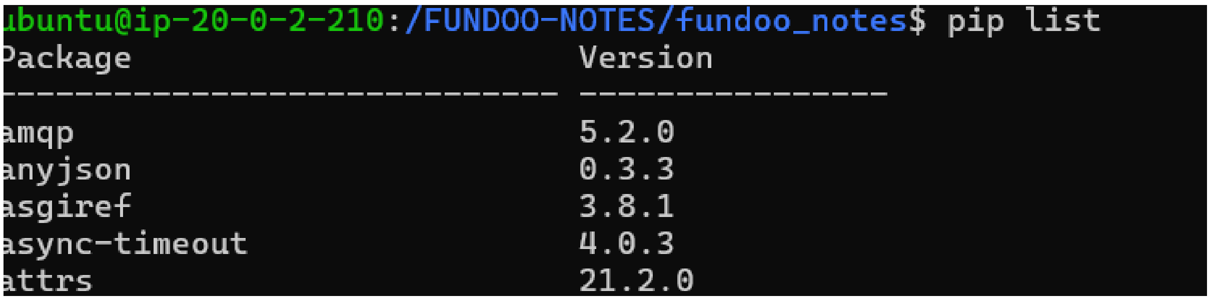

```
ubuntu@ip-20-0-2-105:~$  python3 -m venv myenv
ubuntu@ip-20-0-2-105:~$ ls
key.txt  myenv
```

Step 3: Activate Virtual Environment

$ source myenv/bin/activate

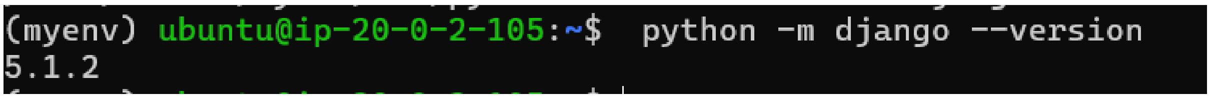

```
ubuntu@ip-20-0-2-105:~$  source myenv/bin/activate
(myenv) ubuntu@ip-20-0-2-105:~$ 
```

Step 4: pip install -r requirements.txt


```
ubuntu@ip-20-0-2-210:/FUNDOO-NOTES/fundoo_notes$ pip list
Package                        Version
------------------------------ ----------------
amqp                           5.2.0
anyjson                        0.3.3
asgiref                        3.8.1
async-timeout                  4.0.3
attrs                          21.2.0
```

Step 5: Verify Django Version

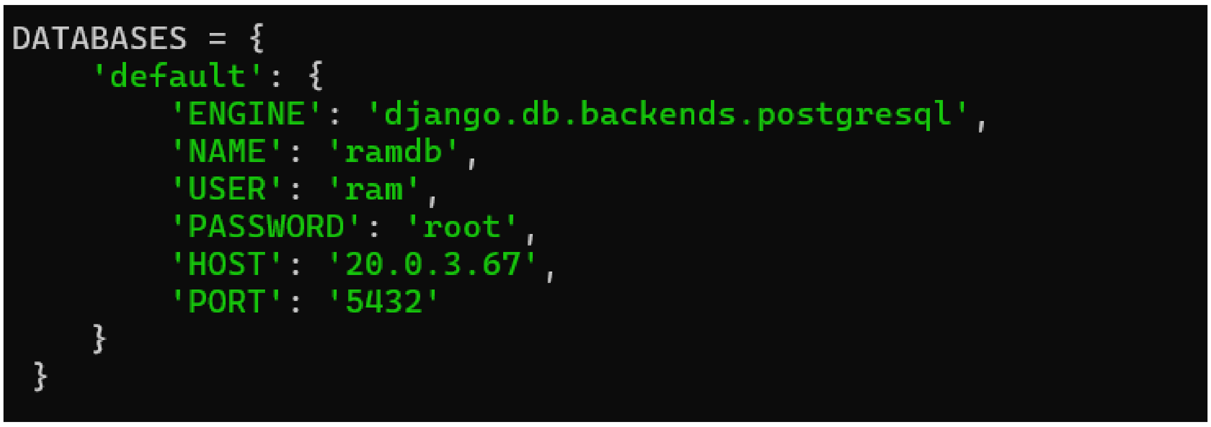
$ python -m django --version



```
(myenv) ubuntu@ip-20-0-2-105:~$  python -m django --version
5.1.2
```

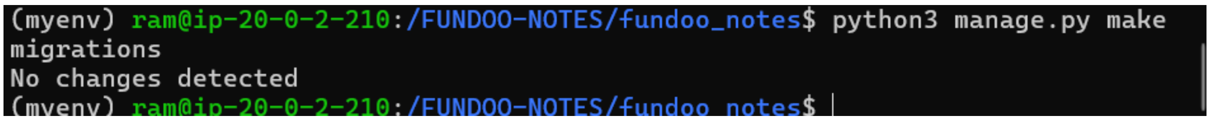
sudo nano /etc/fundoo/env.confg

sudo nano settings.py



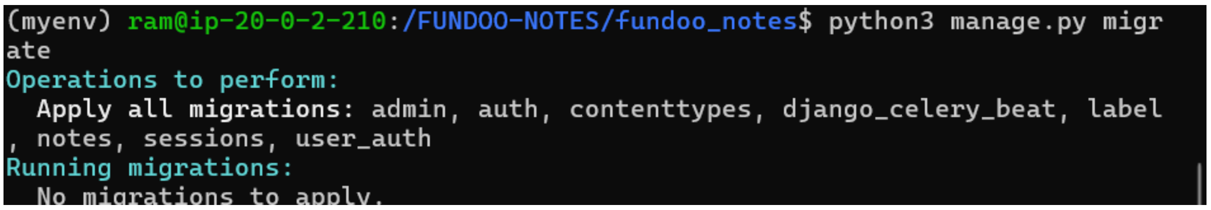
```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'ramdb',
        'USER': 'ram',
        'PASSWORD': 'root',
        'HOST': '20.0.3.67',
        'PORT': '5432'
    }
 }
```

Python3 manage.py makemigrations



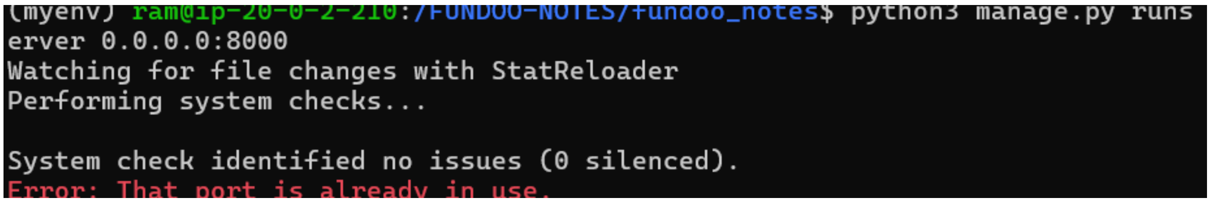
```
(myenv) ram@ip-20-0-2-210:/FUNDOO-NOTES/fundoo_notes$ python3 manage.py make
migrations
No changes detected
(myenv) ram@ip-20-0-2-210:/FUNDOO-NOTES/fundoo_notes$
```

Python3 manage.py migrate



```
(myenv) ram@ip-20-0-2-210:/FUNDOO-NOTES/fundoo_notes$ python3 manage.py migr
ate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, django_celery_beat, label
, notes, sessions, user_auth
Running migrations:
  No migrations to apply.
```

Python3 manage.py runserver 0.0.0.0:8000



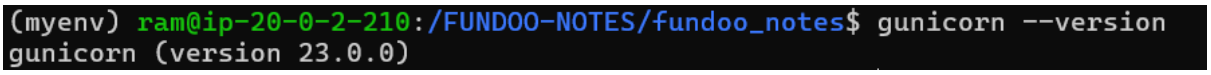
```
(myenv) ram@ip-20-0-2-210:/FUNDOO-NOTES/fundoo_notes$ python3 manage.py runs
erver 0.0.0.0:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
Error: That port is already in use.
```

To install gunicorn

pip install gunicorn

pip show gunicorn

gunicorn --version

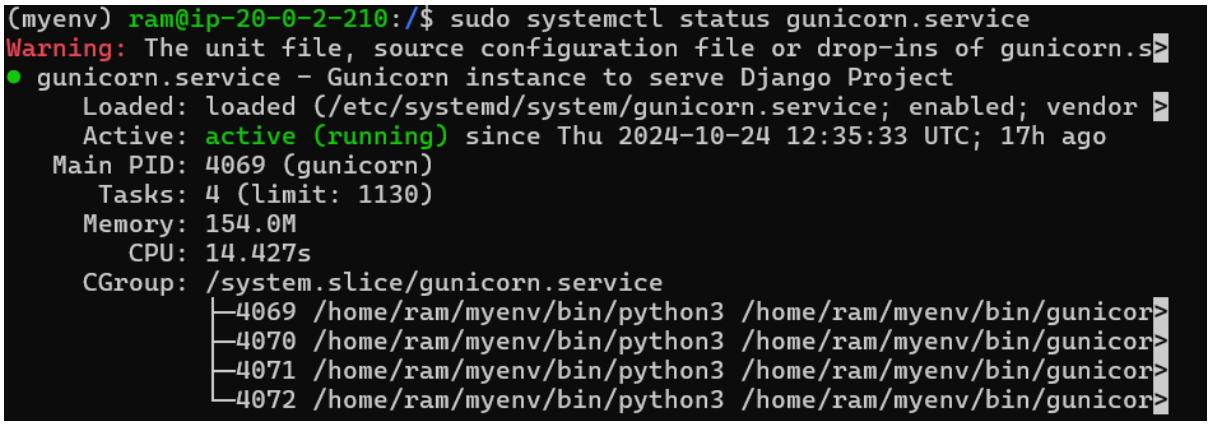


```
(myenv) ram@ip-20-0-2-210:/FUNDOO-NOTES/fundoo_notes$ gunicorn --version
gunicorn (version 23.0.0)
```

sudo systemctl start gunicorn.service

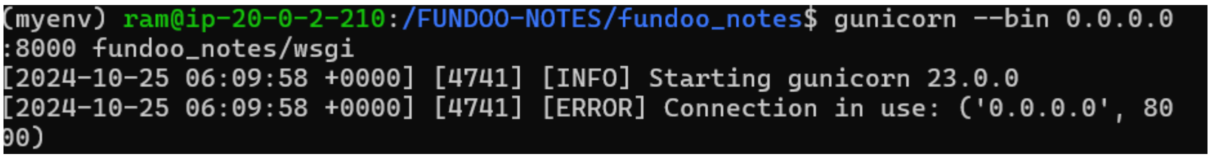
sudo systemctl enable gunicorn.service

sudo systemctl restart gunicorn.service



```
(myenv) ram@ip-20-0-2-210:/$ sudo systemctl status gunicorn.service
Warning: The unit file, source configuration file or drop-ins of gunicorn.s>
● gunicorn.service - Gunicorn instance to serve Django Project
     Loaded: loaded (/etc/systemd/system/gunicorn.service; enabled; vendor >
     Active: active (running) since Thu 2024-10-24 12:35:33 UTC; 17h ago
   Main PID: 4069 (gunicorn)
      Tasks: 4 (limit: 1130)
     Memory: 154.0M
        CPU: 14.427s
     CGroup: /system.slice/gunicorn.service
             ├─4069 /home/ram/myenv/bin/python3 /home/ram/myenv/bin/gunicor>
             ├─4070 /home/ram/myenv/bin/python3 /home/ram/myenv/bin/gunicor>
             ├─4071 /home/ram/myenv/bin/python3 /home/ram/myenv/bin/gunicor>
             └─4072 /home/ram/myenv/bin/python3 /home/ram/myenv/bin/gunicor>
```

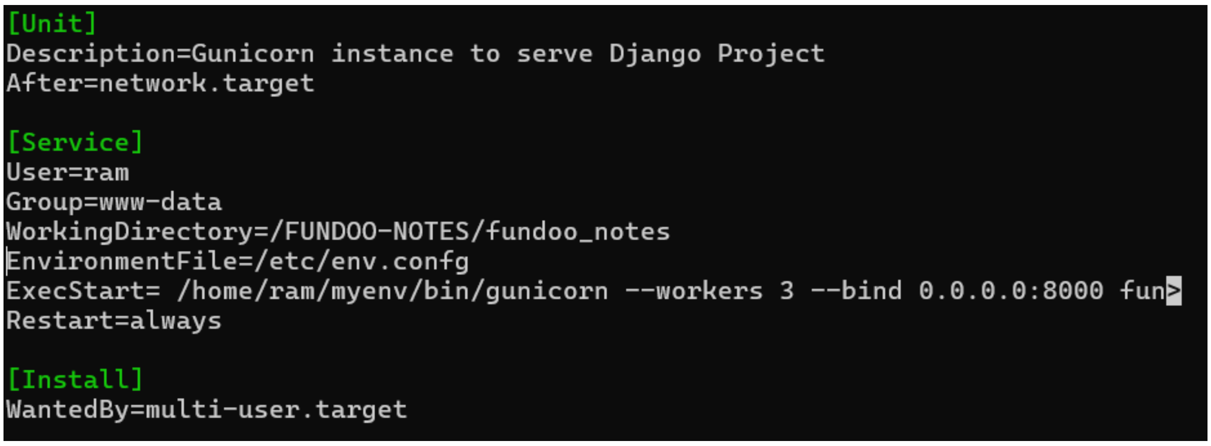
gunicorn --bin 0.0.0.0:8000 fundoo_notes/wsgi



```
(myenv) ram@ip-20-0-2-210:/FUNDOO-NOTES/fundoo_notes$ gunicorn --bin 0.0.0.0
:8000 fundoo_notes/wsgi
[2024-10-25 06:09:58 +0000] [4741] [INFO] Starting gunicorn 23.0.0
[2024-10-25 06:09:58 +0000] [4741] [ERROR] Connection in use: ('0.0.0.0', 80
00)
```

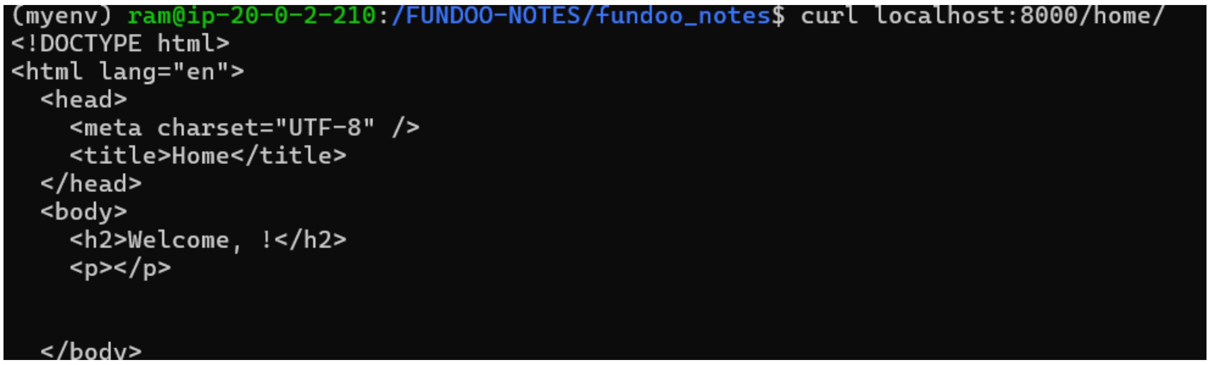
Create a .service file

Sudo nano /etc/systemd/system/fundoo.service



```
[Unit]
Description=Gunicorn instance to serve Django Project
After=network.target

[Service]
User=ram
Group=www-data
WorkingDirectory=/FUNDOO-NOTES/fundoo_notes
EnvironmentFile=/etc/env.confg
ExecStart= /home/ram/myenv/bin/gunicorn --workers 3 --bind 0.0.0.0:8000 fun>
Restart=always

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

To test : Curl localhost:8000/home/



```
(myenv) ram@ip-20-0-2-210:/FUNDOO-NOTES/fundoo_notes$ curl localhost:8000/home/
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Home</title>
  </head>
  <body>
    <h2>Welcome, !</h2>
    <p></p>


  </body>
```

Reference link : https://ultahost.com/knowledge-base/install-django-ubuntu/

Database: Install and configure PostgreSQL.

Step 1: Install PostgreSQL

```
sudo apt install -y postgresql-common -y
sudo systemctl restart postgresql
sudo systemctl status  postgresql
```

Reference link : https://docs.vultr.com/how-to-install-postgresql-on-ubuntu-24-04

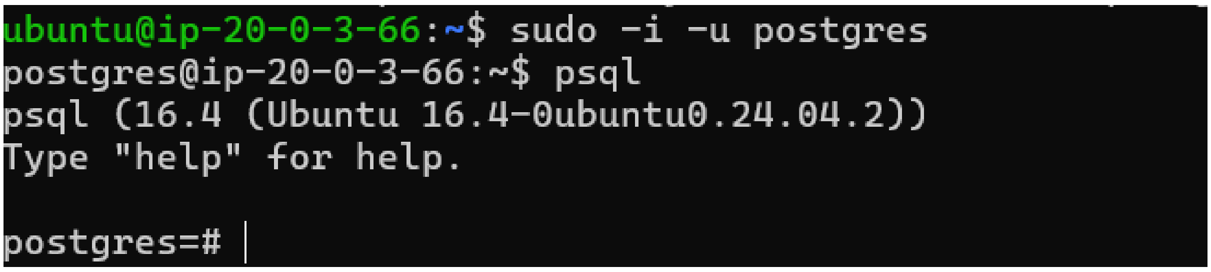
Check postgres is active or not

```
Sudo systemctl start postgres
Sudo systemctl enable postgres
Sudo systemctl status postgres
```

Enter into postgres shell

Step-1 : sudo -I -u postgres

Step-2 psql

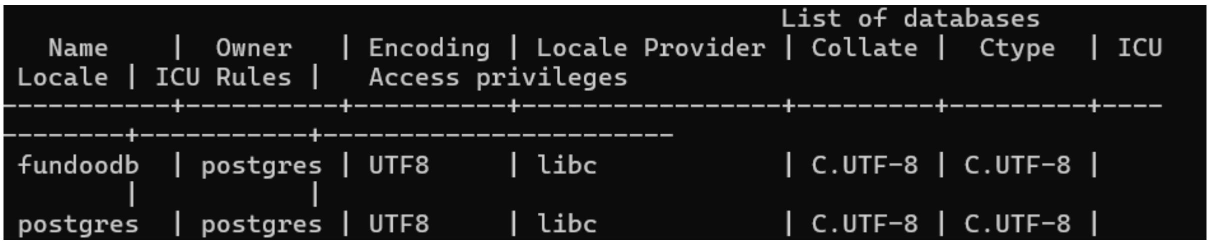


```
ubuntu@ip-20-0-3-66:~$ sudo -i -u postgres
postgres@ip-20-0-3-66:~$ psql
psql (16.4 (Ubuntu 16.4-0ubuntu0.24.04.2))
Type "help" for help.

postgres=#
```

Create an database

Crate database fundooDB;

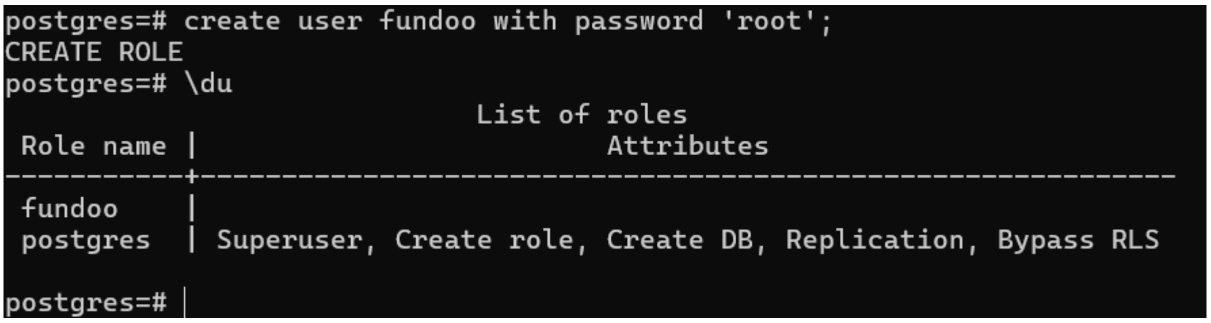
\l



```
                                                   List of databases
   Name    |  Owner   | Encoding | Locale Provider | Collate |  Ctype  | ICU
 Locale | ICU Rules |   Access privileges
-----------+----------+----------+-----------------+---------+---------+----
--------+-----------+-----------------------
 fundoodb  | postgres | UTF8     | libc            | C.UTF-8 | C.UTF-8 |
        |           |
 postgres  | postgres | UTF8     | libc            | C.UTF-8 | C.UTF-8 |
```

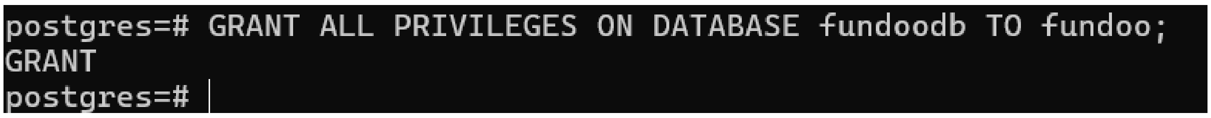
Crate user fundoo with password 'root';

\du



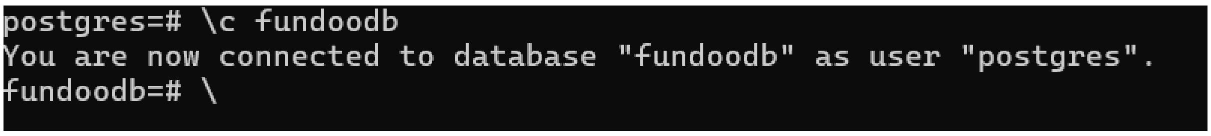
```
postgres=# create user fundoo with password 'root';
CREATE ROLE
postgres=# \du
                             List of roles
 Role name |                         Attributes
-----------+------------------------------------------------------------
 fundoo    |
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS

postgres=#
```

GRANT ALL PRIVILEGES ON DATABASE fundoodb TO fundoo;



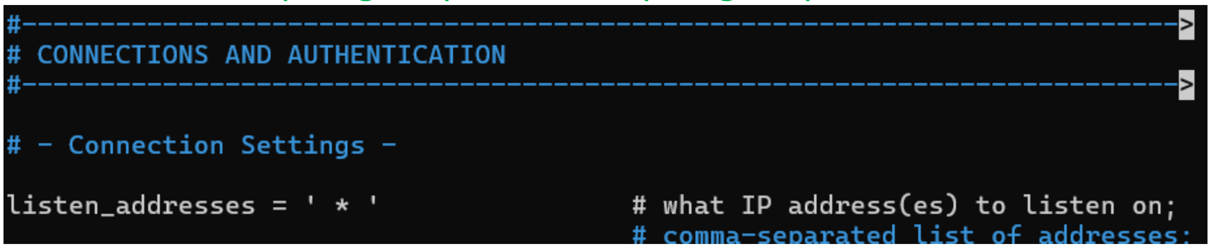
```
postgres=# GRANT ALL PRIVILEGES ON DATABASE fundoodb TO fundoo;
GRANT
postgres=#
```

\c fundoodb => toswitch to another db



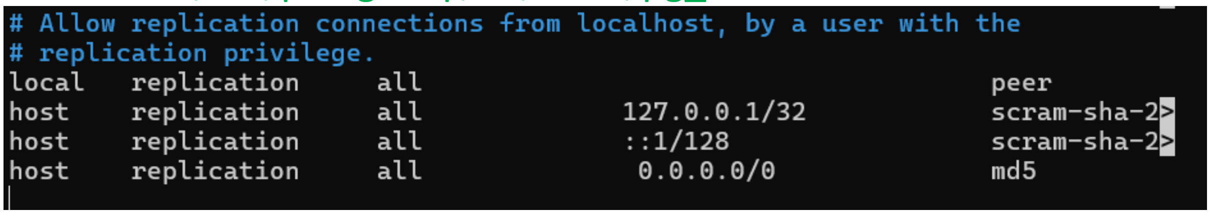
```
postgres=# \c fundoodb
You are now connected to database "fundoodb" as user "postgres".
fundoodb=# \
```

sudo nano /etc/postgresql/16/main/postgresql.conf



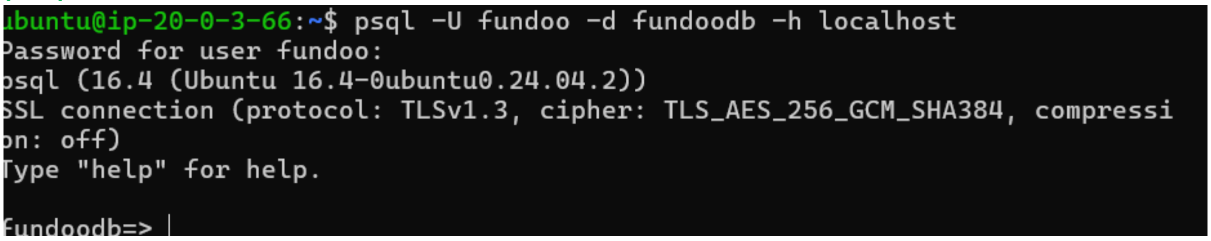
```
#------------------------------------------------------------------------------>
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------>

# - Connection Settings -

listen_addresses = ' * '                # what IP address(es) to listen on;
                                        # comma-separated list of addresses;
```

sudo nano /etc/postgresql/16/main/pg_hba.conf



```
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     peer
host    replication     all             127.0.0.1/32            scram-sha-2>
host    replication     all             ::1/128                 scram-sha-2>
host    replication     all              0.0.0.0/0              md5
```

psql -U fundoo -d fundoodb -h localhost



```
ubuntu@ip-20-0-3-66:~$ psql -U fundoo -d fundoodb -h localhost
Password for user fundoo:
psql (16.4 (Ubuntu 16.4-0ubuntu0.24.04.2))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compressi
on: off)
Type "help" for help.

fundoodb=>
```

complete