3 tier application deployement in VPC infrastructure in AWS

This setup follows a three-tier architecture, with the **database** and **backend** in private subnets and the **frontend** in a public subnet.

Part 1: Set Up the Database (MySQL)

1. **Launch EC2 Instance for MySQL Database**
   - Go to the **EC2 Dashboard** in the AWS Management Console.
   - Click on **Launch Instance**.
   - Choose **Ubuntu Server 22.04 LTS** AMI.
   - Select an appropriate instance type (e.g., t2.micro).
   - Configure the instance:
     - **Network settings**: Choose a **VPC** and a **private subnet** for this instance.
     - **Security Group**: Create a new security group that allows inbound traffic on port 3306 (MySQL) from the private IP of the backend EC2 instance.
2. **Connect to the MySQL EC2 Instance**
   - Connect via SSH using the key pair you specified during instance creation:

```
ssh -i /path/to/your/key.pem ubuntu@<MySQL-EC2-Public-IP>
```

3. **Install MySQL Server**

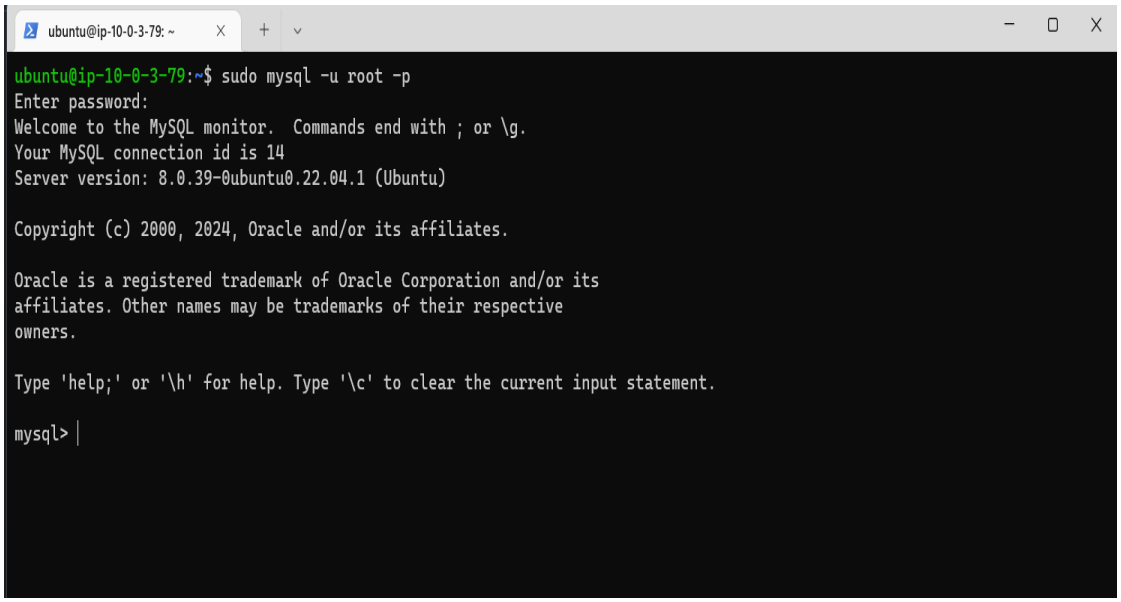
```
sudo apt update
sudo apt install mysql-server -y
```

4. **Secure MySQL Installation**

```
sudo mysql_secure_installation
```

   - Follow the prompts to set up the root password, remove anonymous users, disallow remote root login, and remove the test database.
5. **Create a MySQL Database and User**
   - Log into MySQL as a root user :

sudo mysql -u root -p



- Run the following commands to create the database and user:

  CREATE DATABASE mysqldb;

  CREATE USER 'admin'@'%' IDENTIFIED BY 'admin@12345';

  GRANT ALL PRIVILEGES ON mysqldb.* TO 'admin'@'%';

  FLUSH PRIVILEGES;

  EXIT;

6. **Configure MySQL to Allow Remote Connections**

- Edit the MySQL configuration file:

  sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf

- Find the line:

  bind-address = 127.0.0.1

- Change it to:

  bind-address = 0.0.0.0

- Restart MySQL:

  sudo systemctl restart mysql

7. **Login into mysql as root user and give priviliges to allow all the instances that are going to get launch in private subnet so that later on there is not going to be any conflict**

   sudo mysql -u root -p

   mysql> SET GLOBAL validate_password_policy = LOW;

   CREATE USER 'admin'@'10.0.3.%' IDENTIFIED BY 'admin@12345';

   GRANT ALL PRIVILEGES ON mysqldb.* TO 'admin'@'10.0.3.%';

   FLUSH PRIVILEGES;

   CREATE USER 'admin'@'10.0.2.%' IDENTIFIED BY 'admin@12345';

   GRANT ALL PRIVILEGES ON mysqldb.* TO 'admin'@'10.0.2.%';

   FLUSH PRIVILEGES;\

8. **Also give priviliges to connect to your backend django instance**

   mysql -u root -p

   mysql> CREATE USER 'admin'@'10.0.2.88' IDENTIFIED BY 'admin@12345';

- **Purpose**: This command creates a new MySQL user named admin that can only connect from the IP address 10.0.2.88 (your backend EC2 instance).
- **Security**: By restricting access to a specific IP, you enhance security by ensuring that this user can only connect from your backend instance, reducing the risk of unauthorized access from other sources.

mysql> GRANT ALL PRIVILEGES ON mysqldb.* TO 'admin'@'10.0.2.88';

Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;

Query OK, 0 rows affected (0.01 sec)

To verify mysql is running we can use command

sudo systemctl status mysql

Best Practices for Database User Permissions

1. **Avoid Using Root User**:
   - It's generally not a good idea to use the MySQL root user for application connections due to security risks.
2. **Create a Dedicated User**:
   - You can create a dedicated MySQL user (like admin in your example) with the necessary privileges only on the specific database the application will access.
3. **Grant Specific Privileges**:
   - When creating the user, grant only the required permissions (e.g., SELECT, INSERT, UPDATE, DELETE) on the specific database. In your case, you already created a user named admin and granted it all privileges on the mysqldb database, which is appropriate.

Part 2: Set Up the Backend (Django)

1. **Launch EC2 Instance for Django Backend**
   - Follow the same steps as for the database instance.
   - Configure the security group to allow:
     - **Type**: SSH, **Port**: 22 (for your IP)
     - **Type**: MySQL, **Port**: 3306 (from your database's security group)
     - **Type**: HTTP, **Port**: 80 (for frontend access)
2. **Connect to the Backend EC2 Instance**

   ssh -i "/root/key/key/backend-django-key.pem" ubuntu@10.0.2.88

3. **Create seprate user and and git clone chatapp in root directory and give required priviliges to the user.**

su chatapp

sudo chown chatapp:chatapp /chatapp

sudo chmod 550 /chatapp

4. **Set Up Your Django Application**
   - Clone your Django application from GitHub:

```
cd ~
git clone https://github.com/peyyala7hills/new_chatapp.git
cd new_chatapp
```

5. **Create a Virtual Environment**

```
python3 -m venv venv
source venv/bin/activate
```

6. **Install Required Packages inside virtual environment**

```
sudo apt update
sudo apt install python3-pip python3-dev libmysqlclient-dev nginx git -y

pip install -r requirements.txt
```

requirements.txt contains libraries and dependencies required by the Django project, such as Django itself, database connectors (e.g., mysqlclient), and any other third-party libraries used by the application.

Remember Note

1. you **do not need** to install requirements.txt on the database instance as Python dependencies are not required there. The database instance is only running MySQL

2. you **do not need** to install requirements.txt on the frontend instance if it's only handling **Nginx**. The frontend instance is serving static files (HTML, CSS, JavaScript) and routing traffic to the backend. It doesn't require Python dependencies.

3. You need to install the Python dependencies listed in requirements.txt **only on the backend instance** where your **Django application** will be running.

requirements.txt contains libraries and dependencies required by the Django project, such as Django itself, database connectors (e.g., mysqlclient), and any other third-party libraries used by the application.

7. **Install Django and MySQL Client**

   pip install django mysqlclient

8. **Configure Django Settings**

   - Edit settings.py in your Django project to ensure it connects to the MySQL database: (don't hardcode your credential here use os.getenv method and set your environment variables in Daemon service file and ~/.bashrc)

```
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.mysql',
    'NAME': os.getenv('DB_NAME', ''),
    'USER': os.getenv('DB_USER', ''),
    'PASSWORD': os.getenv('DB_PASSWORD', ''),
    'HOST': os.getenv('DB_HOST', ''),
    'PORT': os.getenv('DB_PORT', '3306'),  # Default port for MySQL
  }
}
```

9. **Open bashrc and add these credentials**

   nano ~/.bashrc

```
#Database Credentials
export DB_USER=admin
export DB_NAME='mysqldb'
export DB_PASSWORD=admin@12345
export DB_HOST=10.0.3.79  # this is database ip
export DB_PORT=3306
```

10. **Run Migrations**

python3 manage.py migrate

```
(venv) chatapp@ip-10-0-2-88:/new_chatapp/fundoo$ python3 manage.py migrate
/new_chatapp/venv/lib/python3.6/site-packages/psycopg2/__init__.py:144: UserWarning: The psycopg2 wheel package will be
renamed from release 2.8; in order to keep installing from binary please use "pip install psycopg2-binary" instead. For
details see: <http://initd.org/psycopg/docs/install.html#binary-install-from-pypi>.
  """)
System check identified some issues:

WARNINGS:
?: (mysql.W002) MySQL Strict Mode is not set for database connection 'default'
        HINT: MySQL's Strict Mode fixes many data integrity problems in MySQL, such as data truncation upon insertion, b
y escalating warnings into errors. It is strongly recommended you activate it. See: https://docs.djangoproject.com/en/2.
1/ref/databases/#mysql-sql-mode
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, fundooapp, sessions, sites
Running migrations:
  No migrations to apply.
```

(here IF CONNECTION GET FAILED U NEED TO GIVE PRIVILIGES TO ADMIN OF BACKEND INSTANCE SO THAT ONLY admin can connect from the IP address 10.0.2.88 (your backend EC2 instance). You can give priviliges like this .

In the database ec2 instance we login into mysql server as a root user

mysql -u root -p

mysql> CREATE USER 'admin'@'10.0.2.88' IDENTIFIED BY 'admin@12345';

- **Purpose**: This command creates a new MySQL user named admin that can only connect from the IP address 10.0.2.88 (your backend EC2 instance).
- **Security**: By restricting access to a specific IP, you enhance security by ensuring that this user can only connect from your backend instance, reducing the risk of unauthorized access from other sources.

mysql> GRANT ALL PRIVILEGES ON mysqldb.* TO 'admin'@'10.0.2.88';

Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;

- **Security**: By restricting access to a specific IP, you enhance security by ensuring that this user can only connect from your backend instance, reducing the risk of unauthorized access from other sources

11. **Run the Django Development Server**

    - Make sure to bind the server to 0.0.0.0 so it can accept external connections:

python manage.py runserver 0.0.0.0:8000

```
chatapp@ip-10-0-2-88: /new_

(venv) chatapp@ip-10-0-2-88:/new_chatapp/fundoo$ python manage.py runserver 0.0.0.0:8000
/new_chatapp/venv/lib/python3.6/site-packages/psycopg2/__init__.py:144: UserWarning: The psycopg2 wheel package will be
renamed from release 2.8; in order to keep installing from binary please use "pip install psycopg2-binary" instead. For
details see: <http://initd.org/psycopg/docs/install.html#binary-install-from-pypi>.
  """)
/new_chatapp/venv/lib/python3.6/site-packages/psycopg2/__init__.py:144: UserWarning: The psycopg2 wheel package will be
renamed from release 2.8; in order to keep installing from binary please use "pip install psycopg2-binary" instead. For
details see: <http://initd.org/psycopg/docs/install.html#binary-install-from-pypi>.
  """)
Performing system checks...

System check identified no issues (0 silenced).
October 26, 2024 - 09:17:28
Django version 2.1.7, using settings 'fundoo.settings'
Starting ASGI/Channels version 2.1.7 development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

verification

to verify the successful connection has been made from backend to database we can use mysql client
mysql -u admin -p -h 10.0.3.79

Part 3: Set Up the Frontend (Nginx)

1. **Launch EC2 Instance for Frontend**
   - Follow the same steps as for the previous instances.
   - Configure the security group to allow:
     - **Type**: SSH, **Port**: 22 (for your IP)
     - **Type**: HTTP, **Port**: 80 (for public access)
2. **Connect to the Frontend EC2 Instance**
   ssh -i "C:\Users\HP\Downloads\frontend-nginx-pair.pem" ubuntu@13.61.1.214

3. **Install Nginx**

```
sudo apt update
sudo apt install nginx -y
```

4. **Configure Nginx**

   - Create a new Nginx configuration file:

```
sudo nano /etc/nginx/sites-available/new_chatapp
```

   - Add the following configuration, in this path

```
server {
  listen 80;
  server_name _;

  # Serve static files
  location /static/ {
    alias /new_chatapp/fundoo/static/;
  }

  # Proxy pass backend requests to Gunicorn or Django server
  location / {
    proxy_pass http://10.0.2.88:8000;

  }
}
```

5. **if your are seeing welcome to apache page on your frontend ip rather than your website that means it is linked to default page so you can unlink it**

```
cd /etc/nginx/sites-enabled
ls -lrt (it will show which is linked to default)
sudo unlink default
```

6. **Enable the Nginx Configuration**

```
sudo ln -s /etc/nginx/sites-available/new_chatapp /etc/nginx/sites-enabled/
```

7. **Test and Restart Nginx**

sudo nginx -t

sudo systemctl restart nginx

verify nginx is running using below command

sudo systemctl status nginx

8. **Open Your Web Browser**
   - Go to http://<Frontend-EC2. You should see your Django application served via Nginx.

Conclusion

Your three-tier application should now be fully set up, with the MySQL database and Django backend on private subnets and the Nginx frontend on a public subnet. This configuration ensures that your database is secure while allowing your application to be accessed over the web.

NOTE

you need to run these commands after restarting instance as settings.py does not have hardcoded database value so by running these command you are establishing a connection with backend

```
export DB_USER=admin
export DB_NAME='mysqldb'
export DB_PASSWORD=admin@12345

export DB_HOST=10.0.3.79

export DB_PORT=3306

python manage.py runserver 0.0.0.0:8000



echo $DB_NAME

echo $DB_USER

echo $DB_PASSWORD

echo $DB_HOST

echo $DB_PORT
```

Settings.py configuration

```
import os

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': os.getenv('DB_NAME', ''),
        'USER': os.getenv('DB_USER', ''),
        'PASSWORD': os.getenv('DB_PASSWORD', ''),
        'HOST': os.getenv('DB_HOST', ''),
        'PORT': os.getenv('DB_PORT', '3306'),  # Default port for MySQL
    }
}
```

Part 4: Setting Up the Daemon Service

1. Modify the systemd service file

Update the systemd service file with the following structure. The main change here is ensuring that the Environment directive is correctly set and also using EnvironmentFile for additional flexibility.

sudo nano /etc/systemd/system/django-app.service

[Unit]

```
Description=Django Application Service

After=network.target

[Service]

User=chatapp

Group=chatapp

WorkingDirectory=/new_chatapp/fundoo

ExecStart=/new_chatapp/venv/bin/python /new_chatapp/fundoo/manage.py runserver 0.0.0.0:8000

Restart=always


# Specify environment variables directly

Environment="DB_NAME=mysqldb"

Environment="DB_USER=admin"

Environment="DB_PASSWORD=admin@12345"

Environment="DB_HOST=10.0.3.79"

Environment="DB_PORT=3306"


[Install]

WantedBy=multi-user.target
```

Ensure that the Environment lines contain no trailing characters like semicolons (;), which could cause issues.

2.Enable the daemon service file and Reload systemd and restart the service nd check the status

After modifying the service file, reload the systemd configuration and restart the service:

```
sudo systemctl enable django-app.service
```

this enable command is important to keep your server running when u reboot or restart your backend instance.

```
sudo systemctl daemon-reload

sudo systemctl restart django-app.service

sudo systemctl Status django-app.service
```

```
(venv) chatapp@ip-10-0-2-88:/new_chatapp$ sudo systemctl status django-app.service
[sudo] password for chatapp:
● django-app.service - Django Application Service
   Loaded: loaded (/etc/systemd/system/django-app.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2024-10-26 07:02:57 UTC; 1h 50min ago
 Main PID: 781 (python)
    Tasks: 13 (limit: 1104)
   CGroup: /system.slice/django-app.service
           ├─781 /new_chatapp/venv/bin/python /new_chatapp/fundoo/manage.py runserver 0.0.0.0:8000
           └─1017 /new_chatapp/venv/bin/python /new_chatapp/fundoo/manage.py runserver 0.0.0.0:8000

Oct 26 08:51:50 ip-10-0-2-88 python[781]: HTTP POST / 200 [0.00, 10.0.3.186:45500]
Oct 26 08:51:51 ip-10-0-2-88 python[781]: HTTP GET / 200 [0.00, 10.0.3.186:24494]
Oct 26 08:52:14 ip-10-0-2-88 python[781]: HTTP GET / 200 [0.01, 10.0.3.186:19892]
Oct 26 08:52:17 ip-10-0-2-88 python[781]: HTTP GET / 200 [0.00, 10.0.3.186:24494]
Oct 26 08:52:18 ip-10-0-2-88 python[781]: HTTP GET / 200 [0.00, 10.0.2.233:4210]
Oct 26 08:52:21 ip-10-0-2-88 python[781]: HTTP GET / 200 [0.00, 10.0.3.186:45500]
Oct 26 08:52:44 ip-10-0-2-88 python[781]: HTTP GET / 200 [0.01, 10.0.3.186:58522]
Oct 26 08:52:47 ip-10-0-2-88 python[781]: HTTP GET / 200 [0.00, 10.0.3.186:24494]
Oct 26 08:52:48 ip-10-0-2-88 python[781]: HTTP GET / 200 [0.00, 10.0.2.233:13586]
Oct 26 08:52:51 ip-10-0-2-88 python[781]: HTTP GET / 200 [0.00, 10.0.3.186:24494]
(venv) chatapp@ip-10-0-2-88:/new_chatapp$
```

3. Verify if the environment variables are being set

You can check the environment variables inside the systemd service by running:

sudo systemctl show django-app.service --property=Environment

This will display the environment variables set for the django-app.service to ensure that they are correctly set.

4. Check logs

If your Django app is still not recognizing the environment variables, check the logs using:

sudo journalctl -u django-app.service

---

4. Use Django's shell to check environment variables:

You can verify if the environment variables are being set by entering the Django shell:

sudo -u chatapp /new_chatapp/venv/bin/python /new_chatapp/fundoo/manage.py shell

Then, inside the shell, run:

import os

print(os.getenv('DB_NAME'))

print(os.getenv('DB_USER'))

print(os.getenv('DB_PASSWORD'))

print(os.getenv('DB_HOST'))

print(os.getenv('DB_PORT'))

If any of these return None, it means the environment variables aren't being picked up by Django.

Daemon service explanation

[Unit]

Description=Django Application Service

After=network.target

[Service]

User=chatapp

Group=chatapp

WorkingDirectory=/new_chatapp/fundoo

ExecStart=/new_chatapp/venv/bin/python /new_chatapp/fundoo/manage.py runserver 0.0.0.0:8000

Restart=always

Specify environment variables directly

Environment="DB_NAME=mysqldb"

Environment="DB_USER=admin"

Environment="DB_PASSWORD=admin@12345"

Environment="DB_HOST=10.0.3.79"

Environment="DB_PORT=3306"

[Install]

WantedBy=multi-user.target