i have a django application (no backend database everything is stored in a folder called "media") , i have created a aws ec2 instance and i bought a domain (www.senthamiz.in) in go daddy now the requirements are i need to configure nginx and gunicorn (i have never worked on it and no idea what is it) and i need to host it in my domain name in aws ec2

To deploy your Django application on an AWS EC2 instance using Nginx and Gunicorn, and to link it to your domain from GoDaddy, follow these steps:

### Prerequisites:

1. \*\*Django application\*\* ready and tested locally.

2. \*\*AWS EC2 instance\*\* with SSH access.

3. \*\*Domain name\*\* registered with GoDaddy.

4. Basic understanding of the command line.

### Steps:

#### 1. Update and Upgrade Your Server

First, SSH into your EC2 instance and update your package lists:

```sh

sudo apt update

sudo apt upgrade -y

```

#### 2. Install Required Packages

Install necessary packages including Python, pip, and virtualenv:

```sh

sudo apt install python3-pip python3-dev python3-venv nginx git

```

#### 3. Clone Your Django Project

Clone your Django project into the EC2 instance. For example:

```sh

cd /home/ubuntu

git clone https://github.com/yourusername/yourrepository.git

cd yourrepository

```

#### 4. Set Up a Virtual Environment and Install Dependencies

Create a virtual environment and activate it:

```sh

python3 -m venv venv

source venv/bin/activate

```

Install your project's dependencies:

```sh

pip install -r requirements.txt

```

#### 5. Configure Gunicorn

Install Gunicorn:

```sh

pip install gunicorn

```

Create a systemd service file for Gunicorn:

```sh

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

```

Add the following content (replace placeholders with your project details):

```ini

[Unit]

Description=gunicorn daemon

After=network.target

[Service]

User=ubuntu

Group=www-data

WorkingDirectory=/home/ubuntu/yourrepository

ExecStart=/home/ubuntu/yourrepository/venv/bin/gunicorn --workers 3 --bind unix:/home/ubuntu/yourrepository/yourproject.sock yourproject.wsgi:application

[Install]

WantedBy=multi-user.target

```

Start and enable Gunicorn:

```sh

sudo systemctl start gunicorn

sudo systemctl enable gunicorn

```

#### 6. Configure Nginx

Remove the default Nginx configuration file:

```sh

sudo rm /etc/nginx/sites-enabled/default

```

Create a new Nginx configuration file for your project:

```sh

sudo nano /etc/nginx/sites-available/yourproject

```

Add the following content (adjust paths and settings as needed):

```ini

server {

listen 80;

server\_name www.senthamiz.in;

location = /favicon.ico { access\_log off; log\_not\_found off; }

location /static/ {

root /home/ubuntu/yourrepository;

}

location / {

include proxy\_params;

proxy\_pass http://unix:/home/ubuntu/yourrepository/yourproject.sock;

}

}

```

Enable the new configuration by creating a symlink:

```sh

sudo ln -s /etc/nginx/sites-available/yourproject /etc/nginx/sites-enabled

```

Test the Nginx configuration:

```sh

sudo nginx -t

```

If the test is successful, restart Nginx:

```sh

sudo systemctl restart nginx

```

#### 7. Configure GoDaddy Domain

In your GoDaddy account, go to the DNS settings for your domain and add an A record pointing to your EC2 instance's public IP address.

For example:

- \*\*Type\*\*: A

- \*\*Name\*\*: @

- \*\*Value\*\*: [your EC2 instance's public IP address]

- \*\*TTL\*\*: 1 Hour (or default)

#### 8. Update Security Group Settings in AWS

Ensure that your EC2 instance's security group allows HTTP and HTTPS traffic. Go to your EC2 dashboard, select your instance, click on the security group, and add rules for:

- \*\*HTTP\*\*: Port 80

- \*\*HTTPS\*\*: Port 443

#### 9. Verify the Setup

Visit your domain (www.senthamiz.in) to ensure everything is working correctly. Your Django application should now be live and accessible via your custom domain.

### Additional Notes:

- \*\*SSL/TLS\*\*: For a secure HTTPS connection, consider using Let's Encrypt to obtain a free SSL certificate. You can use Certbot for this purpose.

- \*\*Static and Media Files\*\*: Ensure that Nginx is correctly serving your static and media files by configuring the `static` and `media` settings in your Nginx configuration.

- \*\*Debugging\*\*: Check logs for any errors. Gunicorn logs can be found in the systemd journal (`sudo journalctl -u gunicorn`), and Nginx logs are typically located in `/var/log/nginx`.

By following these steps, you should have your Django application successfully deployed on an AWS EC2 instance, served by Gunicorn and Nginx, and accessible via your custom domain from GoDaddy.

To deploy your Django application on an AWS EC2 instance using Nginx and Gunicorn, and to link it to your domain from GoDaddy, follow these steps:

**Prerequisites:**

1. **Django application** ready and tested locally.
2. **AWS EC2 instance** with SSH access.
3. **Domain name** registered with GoDaddy.
4. Basic understanding of the command line.

**Steps:**

**1. Update and Upgrade Your Server**

First, SSH into your EC2 instance and update your package lists:

sh

Copy code

sudo apt update sudo apt upgrade -y

**2. Install Required Packages**

Install necessary packages including Python, pip, and virtualenv:

Shx

Copy code

sudo apt install python3-pip python3-dev python3-venv nginx git

**3. Clone Your Django Project**

Clone your Django project into the EC2 instance. For example:

sh

Copy code

cd /home/ubuntu git clone https://github.com/yourusername/yourrepository.git cd yourrepository

**4. Set Up a Virtual Environment and Install Dependencies**

Create a virtual environment and activate it:

sh

Copy code

python3 -m venv venv source venv/bin/activate

Install your project's dependencies:

sh

Copy code

pip install -r requirements.txt

**5. Configure Gunicorn**

Install Gunicorn:

sh

Copy code

pip install gunicorn

Create a systemd service file for Gunicorn:

sh

Copy code

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

Add the following content (replace placeholders with your project details):

ini

Copy code

[Unit] Description=gunicorn daemon After=network.target [Service] User=ubuntu Group=www-data WorkingDirectory=/home/ubuntu/yourrepository ExecStart=/home/ubuntu/yourrepository/venv/bin/gunicorn --workers 3 --bind unix:/home/ubuntu/yourrepository/yourproject.sock yourproject.wsgi:application [Install] WantedBy=multi-user.target

Start and enable Gunicorn:

sh

Copy code

sudo systemctl start gunicorn sudo systemctl enable gunicorn

**6. Configure Nginx**

Remove the default Nginx configuration file:

sh

Copy code

sudo rm /etc/nginx/sites-enabled/default

Create a new Nginx configuration file for your project:

sh

Copy code

sudo nano /etc/nginx/sites-available/yourproject

Add the following content (adjust paths and settings as needed):

ini

Copy code

server { listen 80; server\_name www.senthamiz.in; location = /favicon.ico { access\_log off; log\_not\_found off; } location /static/ { root /home/ubuntu/yourrepository; } location / { include proxy\_params; proxy\_pass http://unix:/home/ubuntu/yourrepository/yourproject.sock; } }

Enable the new configuration by creating a symlink:

sh

Copy code

sudo ln -s /etc/nginx/sites-available/yourproject /etc/nginx/sites-enabled

Test the Nginx configuration:

sh

Copy code

sudo nginx -t

If the test is successful, restart Nginx:

sh

Copy code

sudo systemctl restart nginx

**7. Configure GoDaddy Domain**

In your GoDaddy account, go to the DNS settings for your domain and add an A record pointing to your EC2 instance's public IP address.

For example:

* **Type**: A
* **Name**: @
* **Value**: [your EC2 instance's public IP address]
* **TTL**: 1 Hour (or default)

**8. Update Security Group Settings in AWS**

Ensure that your EC2 instance's security group allows HTTP and HTTPS traffic. Go to your EC2 dashboard, select your instance, click on the security group, and add rules for:

* **HTTP**: Port 80
* **HTTPS**: Port 443

**9. Verify the Setup**

Visit your domain ([www.senthamiz.in](http://www.senthamiz.in/)) to ensure everything is working correctly. Your Django application should now be live and accessible via your custom domain.

**Additional Notes:**

* **SSL/TLS**: For a secure HTTPS connection, consider using Let's Encrypt to obtain a free SSL certificate. You can use Certbot for this purpose.
* **Static and Media Files**: Ensure that Nginx is correctly serving your static and media files by configuring the **static** and **media** settings in your Nginx configuration.
* **Debugging**: Check logs for any errors. Gunicorn logs can be found in the systemd journal (**sudo journalctl -u gunicorn**), and Nginx logs are typically located in **/var/log/nginx**.

By following these steps, you should have your Django application successfully deployed on an AWS EC2 instance, served by Gunicorn and Nginx, and accessible via your custom domain from GoDaddy.