How to Point Domain and Deploy Django Project using Github on Gunicorn & Nginx Remote Server or VPS

What is Gunicorn ? Gunicorn is a Web Server Gateway Interface (WSGI) which receive requests sent to the Web Server from a Client and forwards them onto the Python applications or Web Frameworks e.g. Flask or Django in order to run the appropriate application code for the request. It basically provides a bridge to communicate between your Web Server and Web Application.

- On Local Machine, Goto Your Project Folder then follow below instruction:
  - Open Terminal
  - Activate Your virtual Env
  - Install Django Extensions Package It will help to clear pyc and cache (Optional)

    ```
    pip install django-extensions
    ```
  - Add Django Extensions Package to INSTALLED_APPS in settings.py File

    ```
    INSTALLED_APPS = (
      ...
      'django_extensions',
      ...
    )
    ```
  - Create requirements.txt File

    ```
    pip freeze > requirements.txt
    ```
  - Deactivate Virtual Env
- Get Access to Remote Server via SSH

```
Syntax:- ssh -p PORT USERNAME@HOSTIP
Example:- ssh -p 1034 raj@216.32.44.12
```

- Verify that all required softwares are installed

```
nginx -v
python --version  OR python3 --version
pip --version
- SQLite is Included with Python
  python -c "import sqlite3; print(sqlite3.sqlite_version)"
git --version
```

- Install Software (If required)

```
sudo apt install nginx
sudo apt install python
sudo apt install python3-pip
sudo apt install git
```

- Install virtualenv

```
pip list
sudo pip install virtualenv
OR
sudo apt install python3-virtualenv
```

- Verify Nginx is Active and Running

```
sudo service nginx status
```

- Verify Web Server Ports are Open and Allowed through Firewall

```
sudo ufw status verbose
```

- Exit from Remote Server

```
exit
```

- Login to Your Domain Provider Website
- Navigate to Manage DNS
- Add Following Records:

Type	Host/Name	Value
A	@	Your Remote Server IP
A	www	Your Remote Server IP
AAAA	@	Your Remote Server IPv6
AAAA	www	Your Remote Server IPv6

- Copy Project from Local Machine to Remote Server or VPS. There are two ways to do it:-
  1. Using Command Prompt
     - On Local Windows Machine Make Your Project Folder a Zip File using any of the software e.g. winzip
     - Open Command Prompt
     - Copy Zip File from Local Windows Machine to Linux Remote Server

     ```
     Syntax:- scp -P Remote_Server_Port Source_File_Path Destination_Path
     Example:- scp -P 1034 miniblog.zip raj@216.32.44.12:
     ```

     - Copied Successfully
     - Get Access to Remote Server via SSH

     ```
     Syntax:- ssh -p PORT USERNAME@HOSTIP
     Example:- ssh -p 1034 raj@216.32.44.12
     ```

     - Unzip the Copied Project Zip File

     ```
     Syntax:- unzip zip_file_name
     Example:- unzip miniblog.zip
     ```

  2. Using Github
     - Open Project on VS Code then Create a .gitignore File (If needed)

- Push your Poject to Your Github Account as Private Repo
- Make Connection between Remote Server and Github Repo via SSH Key
- Generate SSH Keys

```
Syntax:- ssh-keygen -t ed25519 -C "your_email@example.com"
```

- If Permission Denied then Own .ssh then try again to Generate SSH Keys

```
Syntax:- sudo chown -R user_name .ssh
Example:- sudo chown -R raj .ssh
```

- Open Public SSH Keys then copy the key

```
cat ~/.ssh/id_ed25519.pub
```

- Go to Your Github Repo
- Click on Settings Tab
- Click on Deploy Keys option from sidebar
- Click on Add Deploy Key Button and Paste Remote Server's Copied SSH Public Key then Click on Add Key
- Clone Project from your github Repo using SSH Path It requires to setup SSH Key on Github

```
Syntax:- git clone ssh_repo_path
Example:- git clone git@github.com:geekyshow1/miniblog.git
```

- Create Virtual env

```
cd ~/project_folder_name
Syntax:- virtualenv env_name
Example:- virtualenv mb
```

- Activate Virtual env

```
Syntax:- source virtualenv_name/bin/activate
Example:- source mb/bin/activate
```

- Install Dependencies

```
pip install -r requirements.txt
```

- Install Gunicorn

```
pip install gunicorn
```

- Deactivate Virtualenv

```
deactivate
```

- Create System Socket File for Gunicorn

```
Syntax:- sudo nano /etc/systemd/system/your_domain.gunicorn.socket
Example:- sudo nano /etc/systemd/system/sonamkumari.com.gunicorn.socket
```

- Write below code inside sonamkumari.com.gunicorn.socket File

```
Syntax:-
[Unit]
```

```
Description=your_domain.gunicorn socket

[Socket]
ListenStream=/run/your_domain.gunicorn.sock

[Install]
WantedBy=sockets.target

Example:-
[Unit]
Description=sonamkumari.com.gunicorn socket

[Socket]
ListenStream=/run/sonamkumari.com.gunicorn.sock

[Install]
WantedBy=sockets.target
```

- Create System Service File for Gunicorn

```
Syntax:- sudo nano /etc/systemd/system/your_domain.gunicorn.service
Example:- sudo nano /etc/systemd/system/sonamkumari.com.gunicorn.service
```

- Write below code inside sonamkumari.com.gunicorn.service File

```
Syntax:-
[Unit]
Description=your_domain.gunicorn daemon
Requires=your_domain.gunicorn.socket
After=network.target

[Service]
User=username
Group=groupname
WorkingDirectory=/home/username/project_folder_name
ExecStart=/home/username/project_folder_name/virtual_env_name/bin/gunicorn \
          --access-logfile - \
          --workers 3 \
          --bind unix:/run/your_domain.gunicorn.sock \
          inner_project_folder_name.wsgi:application

[Install]
WantedBy=multi-user.target

Example:-
[Unit]
Description=sonamkumari.com.gunicorn daemon
Requires=sonamkumari.com.gunicorn.socket
```

```
After=network.target

[Service]
User=raj
Group=raj
WorkingDirectory=/home/raj/miniblog
ExecStart=/home/raj/miniblog/mb/bin/gunicorn \
          --access-logfile - \
          --workers 3 \
          --bind unix:/run/sonamkumari.com.gunicorn.sock \
          miniblog.wsgi:application

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

- Start Gunicorn Socket and Service

```
Syntax:- sudo systemctl start your_domain.gunicorn.socket
Example:- sudo systemctl start sonamkumari.com.gunicorn.socket

Syntax:- sudo systemctl start your_domain.gunicorn.service
Example:- sudo systemctl start sonamkumari.com.gunicorn.service
```

- Enable Gunicorn Socket and Service

```
Syntax:- sudo systemctl enable your_domain.gunicorn.socket
Example:- sudo systemctl enable sonamkumari.com.gunicorn.socket

Syntax:- sudo systemctl enable your_domain.gunicorn.service
Example:- sudo systemctl enable sonamkumari.com.gunicorn.service
```

- Check Gunicorn Status

```
sudo systemctl status sonamkumari.com.gunicorn.socket
sudo systemctl status sonamkumari.com.gunicorn.service
```

- Restart Gunicorn (You may need to restart everytime you make change in your project code)

```
sudo systemctl daemon-reload
sudo systemctl restart sonamkumari.com.gunicorn
```

- Create Virtual Host File

```
Syntax:- sudo nano /etc/nginx/sites-available/your_domain
Example:- sudo nano /etc/nginx/sites-available/sonamkumari.com
```

- Write following Code in Virtual Host File

```
Syntax:-
server{
    listen 80;
```

```
    listen [::]:80;

    server_name your_domain www.your_domain;

    location = /favicon.ico { access_log off; log_not_found off; }

    location / {
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_pass http://unix:/run/your_domain.gunicorn.sock;
    }

    location  /static/ {
        root /var/www/project_folder_name;
    }

    location  /media/ {
        root /var/www/project_folder_name;
    }
}

Example:-
server{
    listen 80;
    listen [::]:80;

    server_name sonamkumari.com www.sonamkumari.com;

    location = /favicon.ico { access_log off; log_not_found off; }

    location / {
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_pass http://unix:/run/sonamkumari.com.gunicorn.sock;
    }

    location  /static/ {
        root /var/www/miniblog;
    }

    location  /media/ {
        root /var/www/miniblog;
```

```
    }
}
```

- Enable Virtual Host or Create Symbolic Link of Virtual Host File

```
Syntax:- sudo ln -s /etc/nginx/sites-available/virtual_host_file /etc/nginx/sites-enabled/vi
Example:- sudo ln -s /etc/nginx/sites-available/sonamkumari.com /etc/nginx/sites-enabled/son
```

- Check Configuration is Correct or Not

```
sudo nginx -t
```

- Restart Nginx

```
sudo service nginx restart
```

- Fix Error:DisallowedHost at / Invalid HTTP_HOST header:
  - Open Django Project settings.py

  ```
  cd ~/project_folder_name/inner_project_folder_name
  nano settings.py
  ```

  - Make below changes

  ```
  ALLOWED_HOST = ["your_domain"]

  Example:-
  ALLOWED_HOST = ["sonamkumari.com", "www.sonamkumari.com"]
  ```

  - Restart Gunicorn (You need to restart everytime you make change in your project code)

  ```
  sudo systemctl daemon-reload
  sudo systemctl restart sonamkumari.com.gunicorn
  ```
- Create required Directories inside /var/www We will use it to serve static and media files only

```
cd /var/www
sudo mkdir project_folder_name
cd project_folder_name
sudo mkdir static media
```

- Make User, Owner of /var/www/project_folder_name

```
cd /var/www
Syntax:- sudo chown -R user:user project_folder_name
Example:- sudo chown -R raj:raj miniblog
```

- If we want to use Development's Media Files then We should move development's media files to public directory (Optional)

```
cd ~/project_folder_name
Syntax:- sudo mv media/* /var/www/project_folder_name/media/
Example:- sudo mv media/* /var/www/miniblog/media/
```

- Open Django Project settings.py

```
cd ~/project_folder_name/inner_project_folder_name
nano settings.py
```

- Make below changes

```
DEBUG = False

STATIC_URL = 'static/'
STATIC_ROOT = "/var/www/miniblog/static/"

MEDIA_URL = '/media/'
MEDIA_ROOT = "/var/www/miniblog/media/"
```

- Restart Gunicorn (You need to restart everytime you make change in your project code)

```
sudo systemctl daemon-reload
sudo systemctl restart sonamkumari.com.gunicorn
```

- Activate Virtual Env

```
cd ~/project_folder_name
source virtualenv_name/bin/activate
```

- Clear pyc Files and Cache. It requires django-extensions package.

```
python manage.py clean_pyc
python manage.py clear_cache
```

- Serve Static Files

```
python manage.py collectstatic
```

- Create Database Tables

```
python manage.py makemigrations
python manage.py migrate
```

- Create Superuser

```
python manage.py createsuperuser
```

- If needed Deactivate Virtual env

```
deactivate
```

- Restart Gunicorn (You may need to restart everytime you make change in your project code)

```
sudo systemctl daemon-reload
sudo systemctl restart sonamkumari.com.gunicorn
```

- Restart Nginx

```
sudo service nginx restart
```

- Now you can make some changes in your project local development VS Code and Pull it on Remote Server (Only if you have used Github)
- Pull the changes from github repo

```
git pull
```

- Restart Gunicorn (You may need to restart everytime you make change in your project code)

```
sudo systemctl daemon-reload
sudo systemctl restart sonamkumari.com.gunicorn
```

How to Automate Django Deployment using Github Action

- On Your Local Machine, Open Your Project using VS Code or any Editor
- Create A Folder named .scripts inside your root project folder e.g. miniblog/.scripts
- Inside .scripts folder Create A file with .sh extension e.g. miniblog/.scripts/deploy.sh
- Write below script inside the created .sh file

```
#!/bin/bash
set -e

echo "Deployment started ..."

# Pull the latest version of the app
echo "Copying New changes...."
git pull origin master
echo "New changes copied to server !"

# Activate Virtual Env
#Syntax:- source virtual_env_name/bin/activate
source mb/bin/activate
echo "Virtual env 'mb' Activated !"

echo "Clearing Cache..."
python manage.py clean_pyc
python manage.py clear_cache

echo "Installing Dependencies..."
pip install -r requirements.txt --no-input

echo "Serving Static Files..."
python manage.py collectstatic --noinput

echo "Running Database migration..."
```

```
python manage.py makemigrations
python manage.py migrate

# Deactivate Virtual Env
deactivate
echo "Virtual env 'mb' Deactivated !"

echo "Reloading App..."
#kill -HUP `ps -C gunicorn fch -o pid | head -n 1`
ps aux |grep gunicorn |grep inner_project_folder_name | awk '{ print $2 }' |xargs kill -HUP

echo "Deployment Finished !"
```

- Go inside .scripts Folder then Set File Permission for .sh File

```
git update-index --add --chmod=+x deploy.sh
```

- Create Directory Path named .github/workflows inside your root project folder e.g. miniblog/.github/workflows
- Inside workflows folder Create A file with .yml extension e.g. miniblog/.github/workflows/deploy.yml
- Write below script inside the created .yml file

```
name: Deploy

# Trigger the workflow on push and
# pull request events on the master branch
on:
  push:
    branches: ["master"]
  pull_request:
    branches: ["master"]

# Authenticate to the the server via ssh
# and run our deployment script
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Deploy to Server
        uses: appleboy/ssh-action@master
        with:
          host: ${{ secrets.HOST }}
          username: ${{ secrets.USERNAME }}
          port: ${{ secrets.PORT }}
          key: ${{ secrets.SSHKEY }}
          script: "cd ~/project_folder_name && ./.scripts/deploy.sh"
```

- Go to Your Github Repo Click on Settings
- Click on Secrets and Variables from the Sidebar then choose Actions
- On Secret Tab, Click on New Repository Secret
- Add Four Secrets HOST, PORT, USERNAME and SSHKEY as below

```
Name: HOST
Secret: Your_Server_IP
```

```
Name: PORT
Secret: Your_Server_PORT
```

```
Name: USERNAME
Secret: Your_Server_User_Name
```

- You can get Server User Name by loging into your server via ssh then run below command

```
whoami
```

- Generate SSH Key for Github Action by Login into Remote Server then run below Command OR You can use old SSH Key But I am creating New one for Github Action

```
Syntax:- ssh-keygen -f key_path -t ed25519 -C "your_email@example.com"
Example:- ssh-keygen -f /home/raj/.ssh/gitaction_ed25519 -t ed25519 -C "gitactionautodep"
```

- Open Newly Created Public SSH Keys then copy the key

```
cat ~/.ssh/gitaction_ed25519.pub
```

- Open authorized_keys File which is inside .ssh/authroized_keys then paste the copied key in a new line

```
cd .ssh
nano authorized_keys
```

- Open Newly Created Private SSH Keys then copy the key, we will use this key to add New Repository Secret On Github Repo

```
cat ~/.ssh/gitaction_ed25519
```

```
Name: SSHKEY
Secret: Private_SSH_KEY_Generated_On_Server
```

- Commit and Push the change to Your Github Repo
- Pull the changes from github to remote server just once this time

```
cd ~/project_folder_name
git pull
```

- Your Deployment should become automate.
- On Local Machine make some changes in Your Project then Commit and Push to Github Repo It will automatically deployed on Live Server
- You can track your action from Github Actions Tab

- If you get any File Permission error in the action then you have to change file permission accordingly.
- All Done