How to Point Domain and Host Django Project using Github on Apache Web Server VPS Hosting

Note - If you get Permission Denied Please use sudo

- 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

- On Local Windows Machine, Goto Your Project Folder then follow below instruction:
- Create a folder in your root project directory then move database file inside this created directory e.g. mbdb/db.sqlite3
- Open settings.py file then change sqlite db file path as it is now inside folder

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'mbdb/db.sqlite3',
    }
}
```

- Save and Close settings.py file
- Open Terminal
- Activate Your virtual Env
- Create requirements.txt File

```
  pip freeze > requirements.txt
```

- Deactivate Virtual Env
- Push your Poject to You Github Account as Private Repo
- To Access Remote Server via SSH

```
Syntax:- ssh -p PORT USERNAME@HOSTIP
Example:- ssh -p 22 root@216.32.44.12
```

Note:- Run Below Commands on Your Remote Server Linux Machine or VPS, Not on Your Local Windows Machine

- Verify that all required softwares are installed

```
  apache2 -v
  python --version
  apache2ctl -M
  pip --version
- SQLite is Included with Python
  python -c "import sqlite3; print(sqlite3.sqlite_version)"
  git --version
```

- If Required Softwares and Modules are not Installed then Install them:

```
sudo apt install apache2
sudo apt install python
sudo apt install libapache2-mod-wsgi-py3
sudo apt install python3-pip
sudo apt install git
```

- Install virtualenv

```
pip list
sudo pip install virtualenv
```

- Verify Apache2 is Active and Running

```
sudo service apache2 status
```

- Verify Web Server Ports are Open and Allowed through Firewall

```
sudo ufw status verbose
```

- Make Connection between Remote Server and Github via SSH Key
- Generate SSH Keys

```
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
- Verify the Connection, Go to Your Server Terminal then run below

```
ssh -T git@github.com
// OR
ssh -vT git@github.com
```

- You may get an error git @ github.com: Permission denied (publickey) If you will try to clone it directly on Web Server Public Folder /var/www So we will clone github repo in User's Home Directory then Move it to Web server Public Directory
- Clone Project from your github account

```
- Using HTTPS Path It doesnt require to setup SSH Key on Github
Syntax:- git clone https_repo_path
Example:- git clone https://github.com/geekyshow1/miniblog.git

- 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
```

- Run ls command to verify that the project is present

```
ls
```

- Move Project Folder to Web Server public directory

```
Syntax:- sudo mv project_folder_name /var/www
Example:- sudo mv miniblog /var/www
```

- Go to Your Project Directory

```
Syntax:- cd /var/www/project_folder_name
Example:- cd /var/www/miniblog
```

- Create Virtual env

```
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
```

- Create Virtual Host File

```
nano /etc/apache2/sites-available/your_domain.conf
```

- Add Following Code in Virtual Host File

```
<VirtualHost *:80>
    ServerName www.example.com
    ServerAdmin contact@example.com
    #Document Root is not required
    #DocumentRoot /var/www/project_folder_name
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    Alias /static /var/www/project_folder_name/static
    <Directory /var/www/project_folder_name/static>
        Require all granted
    </Directory>

    Alias /media /var/www/project_folder_name/media
    <Directory /var/www/project_folder_name/media>
        Require all granted
    </Directory>

    <Directory /var/www/project_folder_name/Inner_project_folder_name>
        <Files wsgi.py>
            Require all granted
        </Files>
    </Directory>

    WSGIDaemonProcess any_name python-home=/var/www/project_folder_name/myprojectenv python-path=/var/www/project_folder_name
    WSGIProcessGroup any_name
    WSGIScriptAlias /  /var/www/project_folder_name/inner_project_folder_name/wsgi.py
</VirtualHost>
```

- Check Configuration is correct or not

```
sudo apache2ctl configtest
```

- Enable Virtual Host

```
cd /etc/apache2/sites-available/
sudo a2ensite your_domain.conf
```

- Restart Apache2

```
sudo service apache2 restart
```

- Open Django Project settings.py

```
cd /var/www/miniblog/miniblog
sudo nano settings.py
```

- Make below changes

```
ALLOWED_HOST = ["your_domain"]
DEBUG = False
STATIC_URL = 'static/'
STATIC_ROOT = BASE_DIR / 'static'
MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'
```

- Restart Apache2

```
sudo service apache2 restart
```

- You can check error logs If you get any error:

```
cd /var/log
su
cd apache2
cat error.log
```

- You can Clear Error Logs (Optional)

```
sudo bash -c 'echo > /var/log/apache2/error.log'
```

- If get Error mod_wsgi (pid=1234): Failed to proxy response from daemon then follow below instructions:
- Open apache2.conf

```
cd /etc/apache2
sudo nano apache2.conf
```

- Write below code in the bottom of apache2.conf file

```
WSGIApplicationGroup %{GLOBAL}
```

- To Know more about % follow this link: https://modwsgi.readthedocs.io/en/develop/configuration-directives/WSGIApplicationGroup.html (https://modwsgi.readthedocs.io/en/develop/configuration-directives/WSGIApplicationGroup.html)
- Restart Apache2

```
sudo service apache2 restart
```

- Serve Static Files

```
cd /var/www/miniblog
python manage.py collectstatic
```

- Create Database Tables

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

- Create Superuser

```
python manage.py createsuperuser
```

- If Database File throws error Permission Denied then Set below permissions
- Make Webserver as owner for database file. Our Webserver is running as www-data and group is also www-data.

```
Syntax:-
sudo chown -R www-data:www-data database_folder
sudo chmod 775 database_folder
sudo chmod 664 database_folder/database_file

Example:-
sudo chown -R www-data:www-data mbdb
sudo chmod 775 mbdb
sudo chmod 664 mbdb/db.sqlite3
```

- If Media Files (User Uploaded Files) throws error Permission Denied then Set below permissions

```
sudo chown -R www-data:www-data media
```

- You may face problem if you work with FTP so to fix this add your user to webserver user group following below instruction:
- Check Your User Group

```
sudo groups raj
```

- Add your User to webserver group

```
sudo usermod -a -G www-data raj
```

- Verify Your User is in Webserver Group

```
sudo groups raj
```

- If needed Deactivate Virtual env

```
deactivate
```

- If you make any changes in your project then you need to pull the new changes from github repo. It will update your website with latest changes.

```
git pull
```

- After making any change in the project you have to either restart server or touch the wsgi file only then it will reflect on website
- Restarting Server is not good idea as there might be multiple website running on same server but still it is solution to reflect changes however we can also use graceful

```
sudo service apache2 restart
// OR
sudo service apache2 graceful
// OR
cd /var/www/miniblog/miniblog
touch wsgi.py
```

Special Tip: If you face error "Name duplicates previous WSGI daemon definition" while installing SSL Certificate for your domain then comment below code then try to install SSL Certificate again and after successful installation un-comment it

```
cd /etc/apache2/sites-available/your_domain.conf

#WSGIDaemonProcess any_name python-home=/var/www/project_folder_name/myprojectenv python-path=/var/www/project_folder_name
#WSGIProcessGroup any_name
#WSGIScriptAlias /  /var/www/project_folder_name/inner_project_folder_name/wsgi.py
```

- A ssl config file will generate. Remember both ssl and non-ssl files can not have same WSGIDaemonProcess and WSGIProcessGroup name (mentioned above as any_name) so you may have to change the name manually in both the files.

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
git pull origin master
echo "New changes copied to server !"

# Activate Virtual Env
source mb/bin/activate
echo "Virtual env 'mb' Activated !"

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 !"

# Reloading Application So New Changes could reflect on website
pushd miniblog
touch wsgi.py
popd

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 /var/www/miniblog && ./.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
- Get Access to Remote Server via SSH

```
Syntax:- ssh -p PORT USERNAME@HOSTIP
Example:- ssh -p 22 root@216.32.44.12
```

- Go to Your Project Directory

```
Syntax:- cd /var/www/project_folder_name
Example:- cd /var/www/miniblog
```

- Pull the changes from github just once this time.

```
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