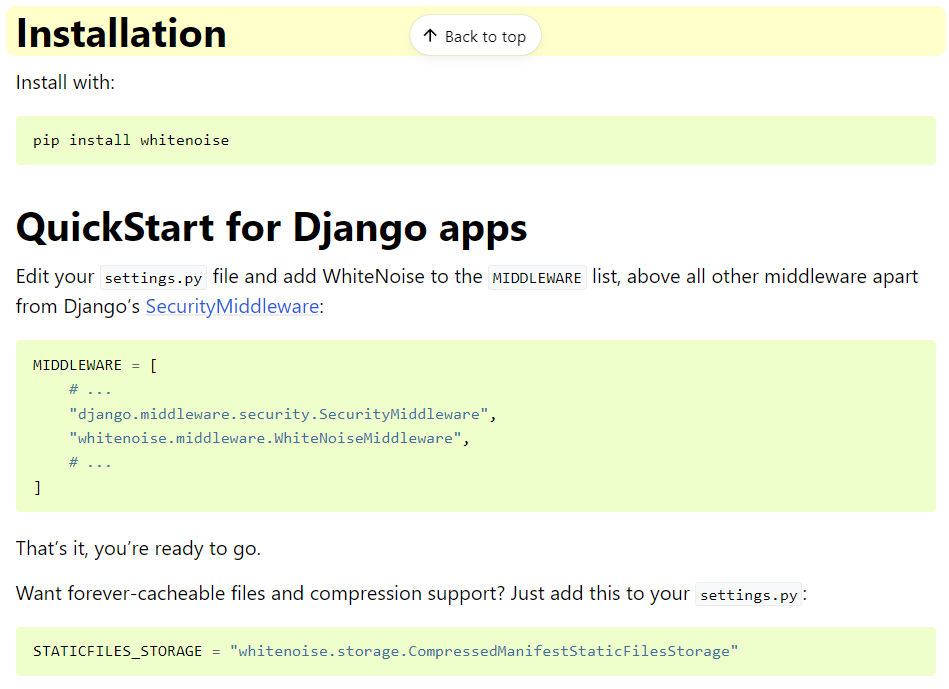
* Please use private git repository, if anyhow public repository is used, then place SECURITY\_KEY in .env file and after cloning the repo to gcp, create .env file using nano editor and place SECURITY\_KEY there. Also please refrain from adding Dockerfile, docker-compose.yml and run.sh files to public git repository.
* Here, in this doc, for domain name examples: example.com, ccstiet.com, escalade.ccstiet.com, etc.
* This doc assumes that already server is set up, if setting up server for first time <https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-16-04>

And run following commands to install necessary packages

sudo apt-get update

sudo apt-get install python3-pip python3-dev nginx docker python-certbot-nginx

1. Go to settings.py file
2. Add your domain name and localhost to ALLOWED\_HOSTS
3. Set DEBUG=False
4. Change ‘NAME’ in ‘DATABASES’ to BASE\_DIR / 'database/db.sqlite3'
5. If using Django4, add CSRF\_TRUSTED\_ORIGINS = ['https://*domain-name*']
6. Add whitenoise



Link for reference: [http://whitenoise.evans.io/en/stable/#installation](http://whitenoise.evans.io/en/stable/%23installation)

In settings.py file

import os

STATIC\_ROOT = os.path.join(BASE\_DIR, 'staticfiles')

Remember whitenoise should also be present in requirements.txt

In project’s urls.py add the content

from django.conf import settings

from django.conf.urls.static import static

Then modify urlpatterns to following

urlpatterns = [

#*your urls……….*

] + static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

1. Add the following to .gitignore

database

staticfiles

privkey.pem

cert.pem

1. First create the following files for the containerization:
2. Dockerfile

Add following contents:

# For more information, please refer to https://aka.ms/vscode-docker-python

FROM python:3.8-slim-buster

# Keeps Python from generating .pyc files in the container

ENV PYTHONDONTWRITEBYTECODE=1

# Turns off buffering for easier container logging

ENV PYTHONUNBUFFERED=1

# Install pip requirements

COPY requirements.txt .

RUN pip install gunicorn[gevent]

RUN python -m pip install -r requirements.txt

WORKDIR /app

COPY . /app

# Creates a non-root user with an explicit UID and adds permission to access the /app folder

# For more info, please refer to https://aka.ms/vscode-docker-python-configure-containers

RUN adduser -u 5678 --disabled-password --gecos "" appuser && chown -R appuser /app

USER root

# During debugging, this entry point will be overridden. For more information, please refer to https://aka.ms/vscode-docker-python-debug

CMD ["sh", "run.sh"]

1. docker-compose.yml

Add following contents:

version: '3.4'

services:

*project-name*:

image: ccs/*project-name*

build:

context: .

dockerfile: ./Dockerfile

ports:

- "6541:6541"

volumes:

- ./database/:/app/database/

restart: always

**Note:** Here please replace *project-name* by your project name. Also, I assume that the repository name is same as project name. (If not then please try it)

1. run.sh

Add following contents:

python manage.py makemigrations

python manage.py migrate

python manage.py collectstatic

gunicorn --worker-class gevent --certfile= cert.pem --keyfile=privkey.pem --bind 0.0.0.0:6541 *django-project-name-folder*.wsgi:application

Here *django-project-name-folder* means the name of folder which contains settings.py and wsgi.py files.

1. SSH to gcp, in user’s folder, clone the project repository.
2. Go to project’s folder
3. Run command

sudo nano /etc/nginx/sites-available/*project-name*

Replace *project-name* with your project name

1. Now nano editor will be opened up, add the following content

server {

listen 80;

server\_name *domain-name*;

return 301 https://$server\_name$request\_uri;

}

server {

listen 443 ssl;

server\_name *domain-name*;

location / {

proxy\_pass https://localhost:6541;

}

}

Replace *domain-name* with domain name

1. Run command

sudo ln -s /etc/nginx/sites-available/*project-name* /etc/nginx/sites-enabled

Please replace *project-name* with your project name

1. Run command:

sudo certbot --nginx -d *domain-name*

Replace *domain-name* with your domain name

If it asks for an option, enter **2**, i.e. Redirect

1. Run command:

sudo cp /etc/letsencrypt/live/*domain-name*/cert.pem /etc/letsencrypt/live/*domain-name*/privkey.pem ./

Replace *domain-name* with domain name

1. Run command:

sudo docker-compose up –build

1. After the container is successfully built, then hit Ctrl + Z
2. Run command

sudo docker-compose up -d –build

1. Run commands

sudo nginx -t

sudo systemctl restart nginx

References:

<https://medium.com/developer-student-clubs-tiet/hosting-multiple-containerized-websites-on-a-single-server-d341e7d38d88>

<https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-ubuntu-16-04>

<https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-16-04>

<https://serverfault.com/questions/67316/in-nginx-how-can-i-rewrite-all-http-requests-to-https-while-maintaining-sub-dom>