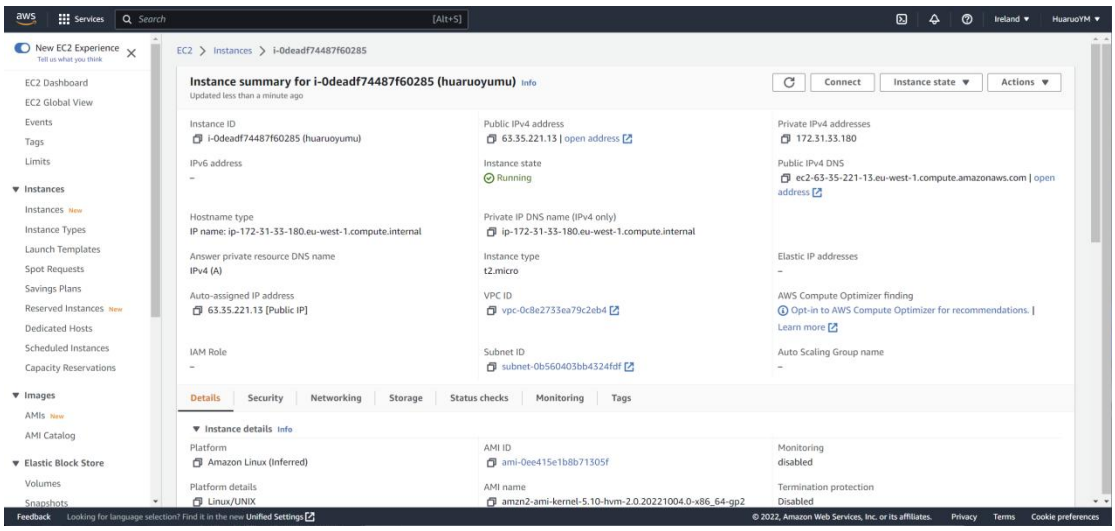
Advanced Web Mapping: Assignment 1

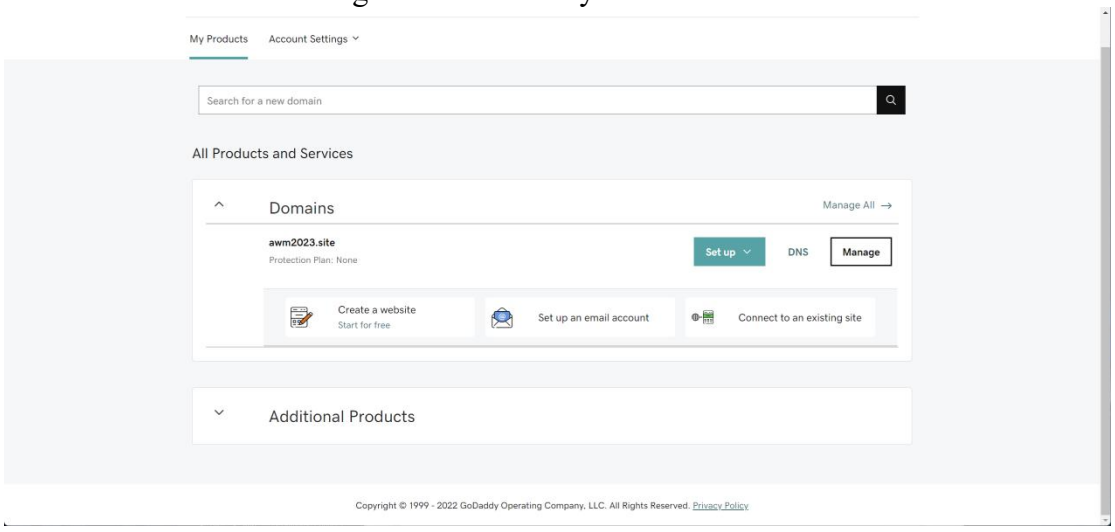
Student number: D18130495
Student name: Yushun Zeng

Website link: https://www.awm2023.site
Pgadmin4 link: https://www.awm2023.site/pgadmin4
Username: D18130495@mytudublin.ie
Password: 123456
Connection:
Host: 63.35.221.13 Port: 25432
Database: gis Username: postgres Password: 123456
Github link: https://github.com/D18130495/Advanced-Web-Mapping

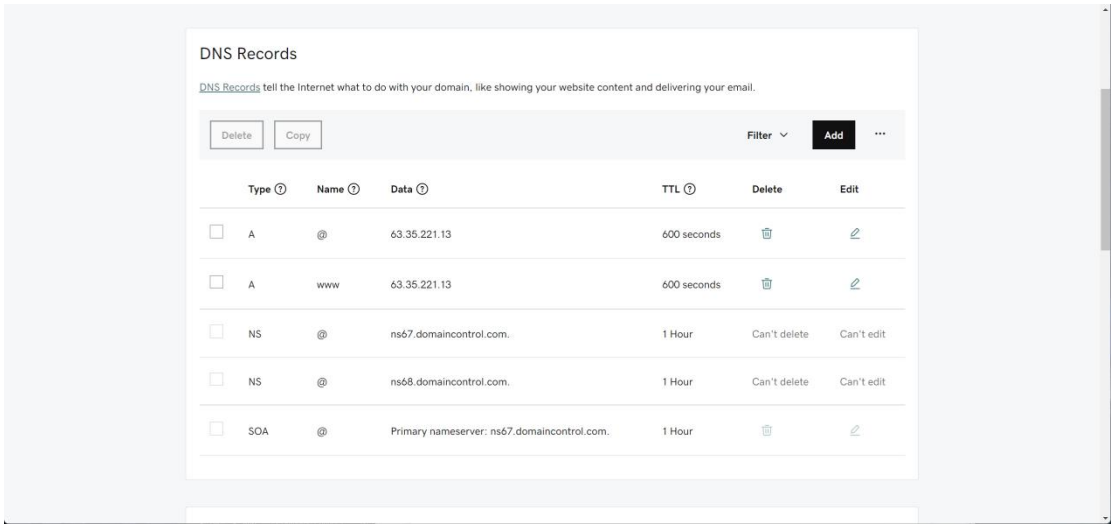
Deploy step:
1. The could server I used is Amazon AWS.



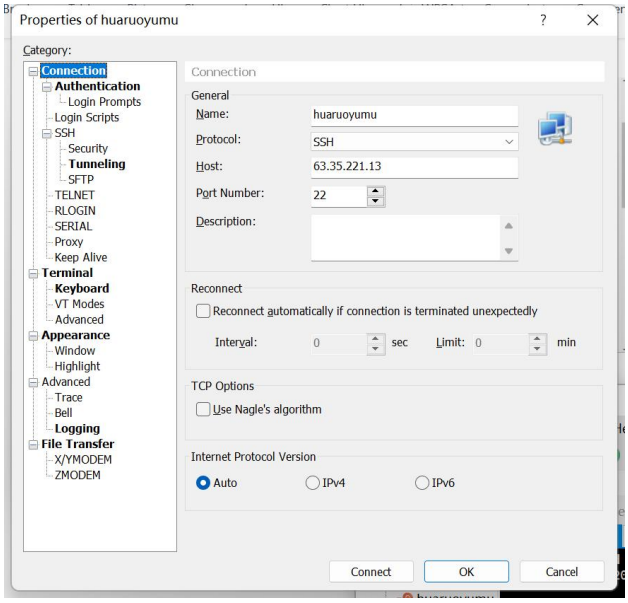
2. The domain name I brought from GoDaddy.

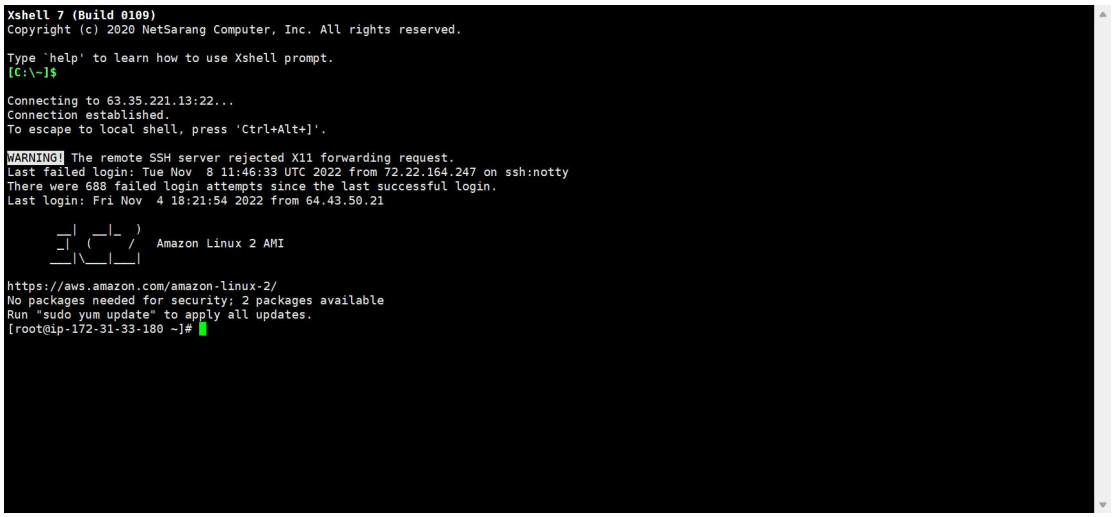


3. Added the domain name’s DNS.

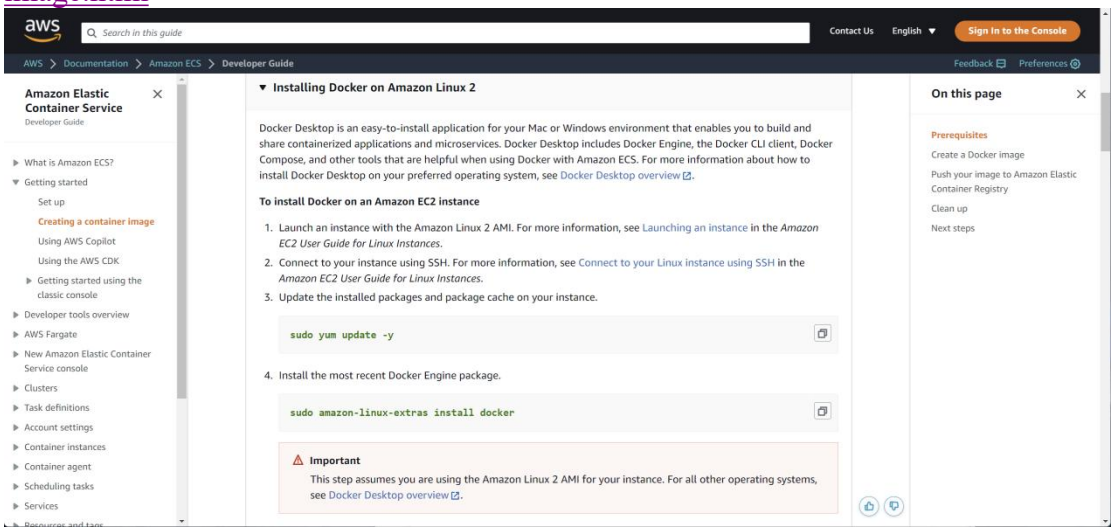


4. Connect to the cloud server var the ssh connection.





5. Install Docker and create network.
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/create-container-image.html



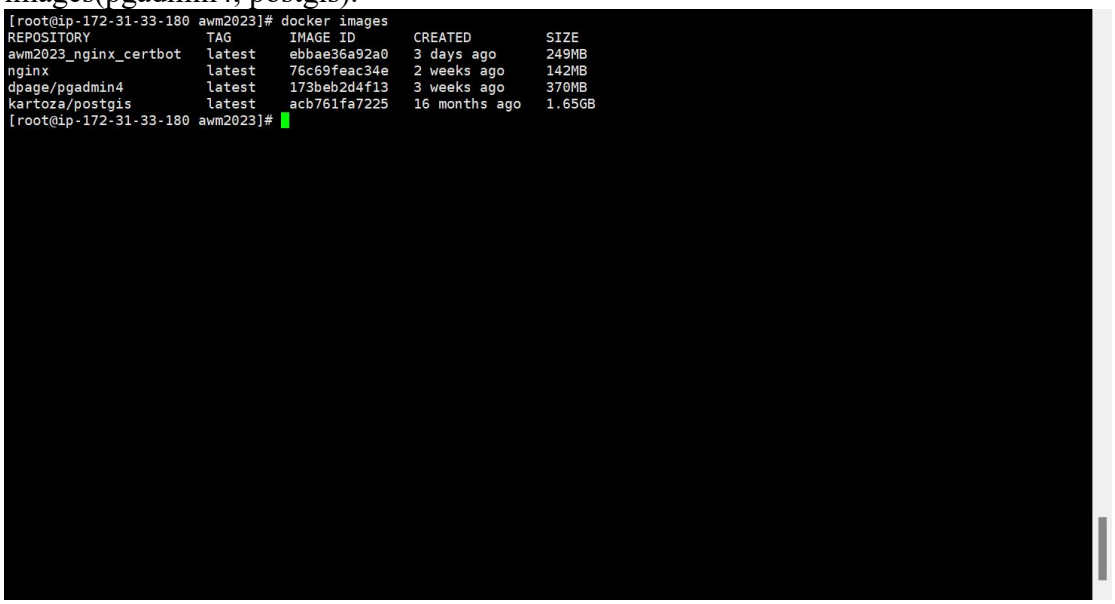
Create new docker network by using docker network create awm2023

```
[root@ip-172-31-33-180 ~]# docker network list
NETWORK ID     NAME      DRIVER    SCOPE
8b3c87a221d1   awm2023   bridge    local
b608fc7e3b58   bridge    bridge    local
b842df81f6f2   host      host      local
1f714291e778   none      null      local
[root@ip-172-31-33-180 ~]#
```

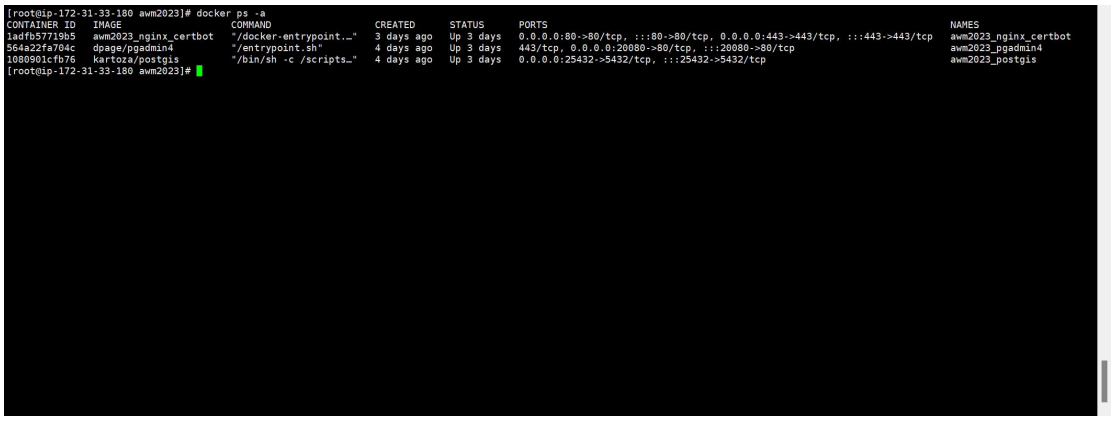
6. Create docker file for get Nginx_Certbot image.

```
[root@ip-172-31-33-180 awm2023]# cat Dockerfile
FROM nginx
MAINTAINER Yushun Zeng
RUN apt-get -y update && apt-get -y upgrade && apt-get -y install software-properties-common certbot python3-certbot-nginx[root@ip-172-31-33-180 awm2023
]#
```

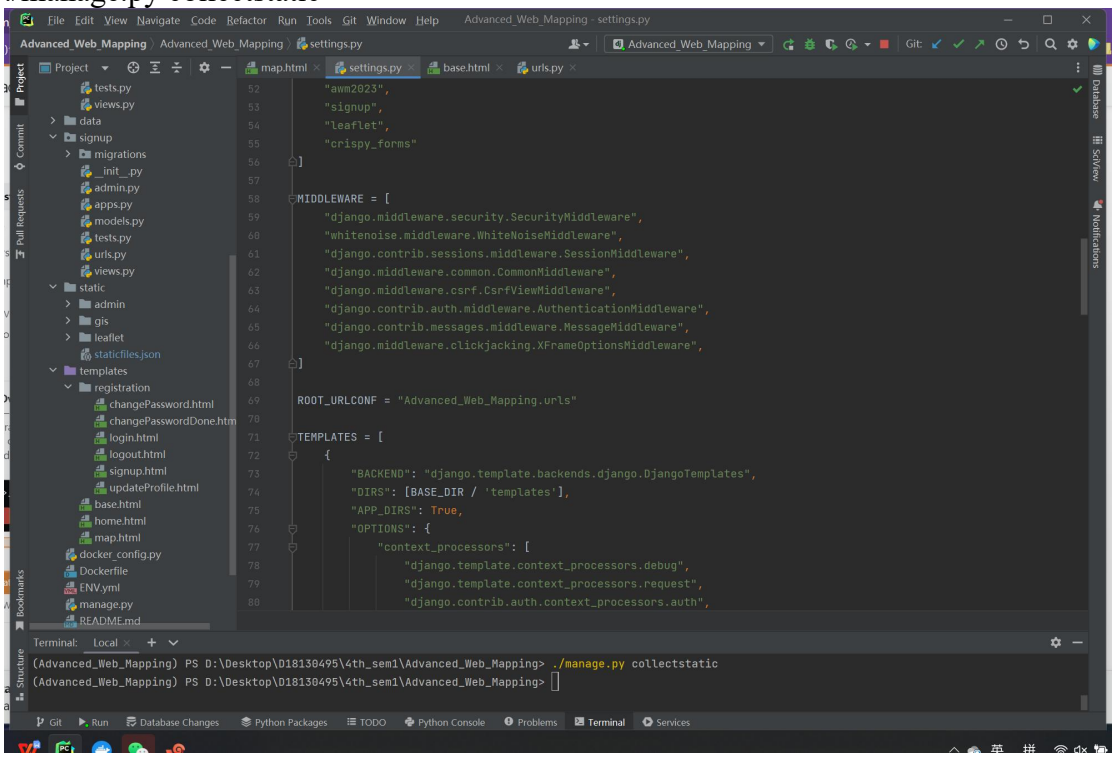
Run the docker file to get Nginx_Certbot image and also get other needed images(pgadmin4, postgis).



Then create the container for Nginx_Certbot, pgadmin4 and postgis under the awm2023 network.



7. Update Django project for deployment.
First, update whitenoise setting and generate static folder by using:
./manage.py collectstatic



Second, update setting.py to detect if the application run on the local host or cloud server and to enable some setting.

```
if socket.gethostname() == "192.168.192.1":
    DATABASES["default"]["HOST"] = "localhost"
    DATABASES["default"]["PORT"] = docker_config.POSTGIS_PORT
else:
    DATABASES["default"]["HOST"] = docker_config.CLOUD_POSTGIS_HOST
    DATABASES["default"]["PORT"] = docker_config.CLOUD_POSTGIS_PORT

# Set DEPLOY_SECURE to True only for LIVE deployment
if docker_config.DEPLOY_SECURE:
    DEBUG = False
    TEMPLATES[0]["OPTIONS"]["debug"] = False
    ALLOWED_HOSTS = ['www.awm2023.site', '63.35.221.13', '127.0.0.1']
    CSRF_COOKIE_SECURE = True
    SESSION_COOKIE_SECURE = True
else:
    DEBUG = True
    TEMPLATES[0]["OPTIONS"]["debug"] = True
    ALLOWED_HOSTS = ['*', ]
    CSRF_COOKIE_SECURE = False
    SESSION_COOKIE_SECURE = False
```

map.html | settings.py | docker_config.py | base.html | urls.py

```
POSTGIS_PORT = 25432
CLOUD_POSTGIS_HOST = "63.35.221.13"
CLOUD_POSTGIS_PORT = 25432
DEPLOY_SECURE = True
```

Third, update SECRET_KEY for the application for security reason.

```
# SECURITY WARNING: keep the secret key used in production secret!
with open('secret_key.txt') as f:
    SECRET_KEY = f.read().strip()
```

map.html | settings.py | secret_key.txt | base.html | urls.py

```
django-insecure--66)l$3pa!$xk$i^@k=^isi-1qns5sv1y!-9)6@d^o7%f)34k^
```

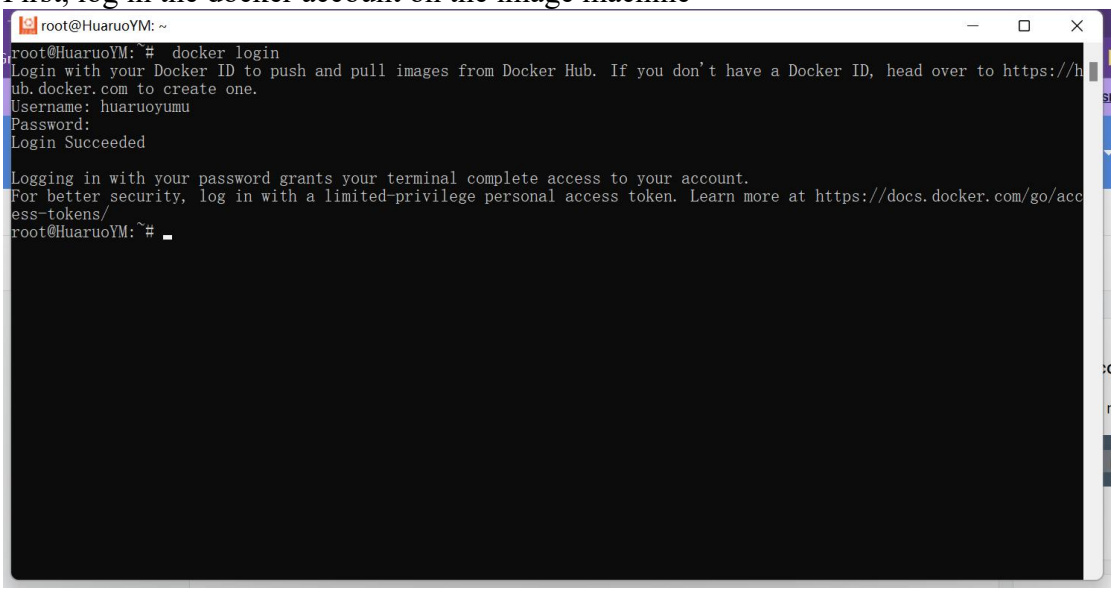
8. Build the project image and push to the Docker Hub.
The docker file that I used to build project image:
https://github.com/D18130495/Advanced-Web-Mapping/blob/main/Dockerfile

Build project image by using docker build -t huaruoyumu/awm2023 .

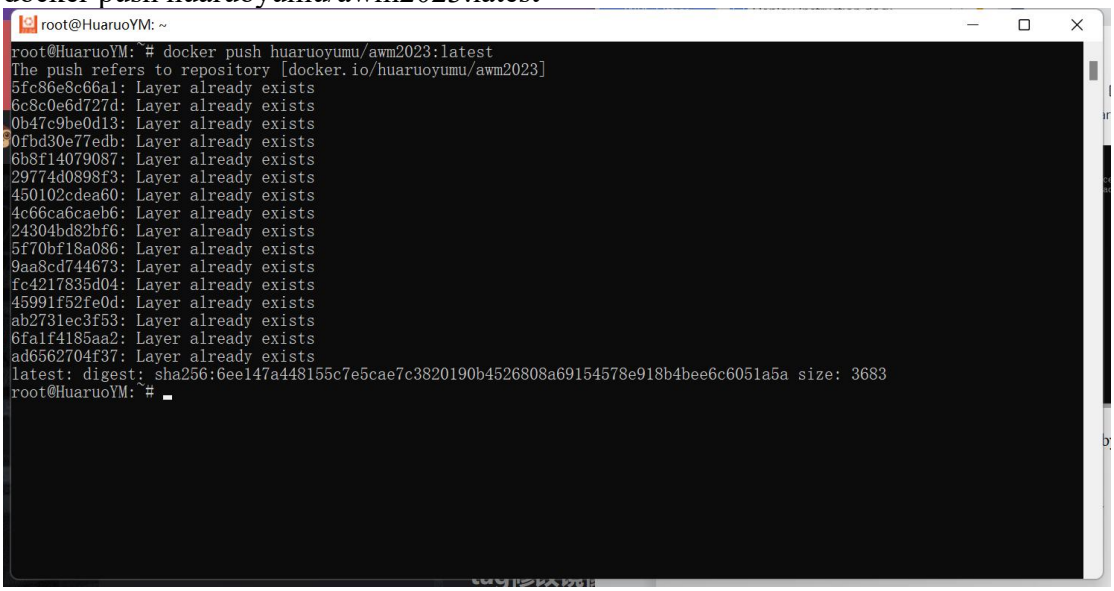
```
[+] Building 10.9s (17/18)
[+] Building 14.8s (19/19) FINISHED
 => [internal] load build definition from Dockerfile                                                                0.1s
 => => transferring dockerfile: 2.00kB                                                                              0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [internal] load metadata for docker.io/continuumio/miniconda3:latest                                            2.6s
 => [ 1/14] FROM docker.io/continuumio/miniconda3@sha256:977263e8d1e476972fddab1c75fe050dd3cd17626390e874448bd92721fd659b  0.0s
 => [internal] load build context                                                                                   0.2s
 => => transferring context: 12.37MB                                                                                0.2s
 => CACHED [ 2/14] RUN apt-get -y update && apt-get -y upgrade                                                      0.0s
 => CACHED [ 3/14] RUN conda update -n base conda && conda update -n base --all                                     0.0s
 => CACHED [ 4/14] RUN mkdir -p /usr/src/app                                                                        0.0s
 => CACHED [ 5/14] WORKDIR /usr/src/app                                                                             0.0s
 => CACHED [ 6/14] RUN apt-get -y install build-essential python3-cffi libcairo2 libpango-1.0-0 libpangocairo-1.0-0 libgdk-pixbuf2.0-0 libffi-dev shar  0.0s
 => CACHED [ 7/14] COPY ENV.yml /usr/src/app                                                                        0.0s
 => CACHED [ 8/14] RUN conda env create -n Advanced_Web_Mapping --file ENV.yml                                      0.0s
 => CACHED [ 9/14] RUN echo "conda activate Advanced_Web_Mapping" >> ~/.bashrc                                      0.0s
 => CACHED [10/14] RUN conda config --add channels conda-forge && conda config --set channel_priority strict       0.0s
 => CACHED [11/14] RUN cat ~/.condarc                                                                               0.0s
 => CACHED [12/14] RUN conda install uwsgi                                                                          0.0s
 => [13/14] COPY . /usr/src/app                                                                                     0.5s
 => [14/14] RUN python manage.py collectstatic --no-input                                                          10.9s
 => exporting to image                                                                                              0.4s
 => => exporting layers                                                                                             0.4s
 => => writing image sha256:e106fa804042a59dfc3e8f4a3a934fdd50f3cb94c2b219c24f071854af97071f                        0.0s
 => => naming to docker.io/library/awm2023                                                                          0.0s
```

```
(Advanced_Web_Mapping) PS D:\Desktop\D18130495\4th_sem1\Advanced_Web_Mapping> docker images
REPOSITORY              TAG       IMAGE ID       CREATED          SIZE
huaruoyumu/awm2023      latest    c8a75926f3a6   6 seconds ago    3.89GB
advanced_web_mapping    latest    64a83950b02e   2 weeks ago      3.86GB
postgres                latest    901a82b310d3   3 weeks ago      377MB
dpage/pgadmin4          latest    94c0924749b6   7 weeks ago      366MB
kartoza/postgis         latest    acb761fa7225   16 months ago    1.65GB
(Advanced_Web_Mapping) PS D:\Desktop\D18130495\4th_sem1\Advanced_Web_Mapping>
```

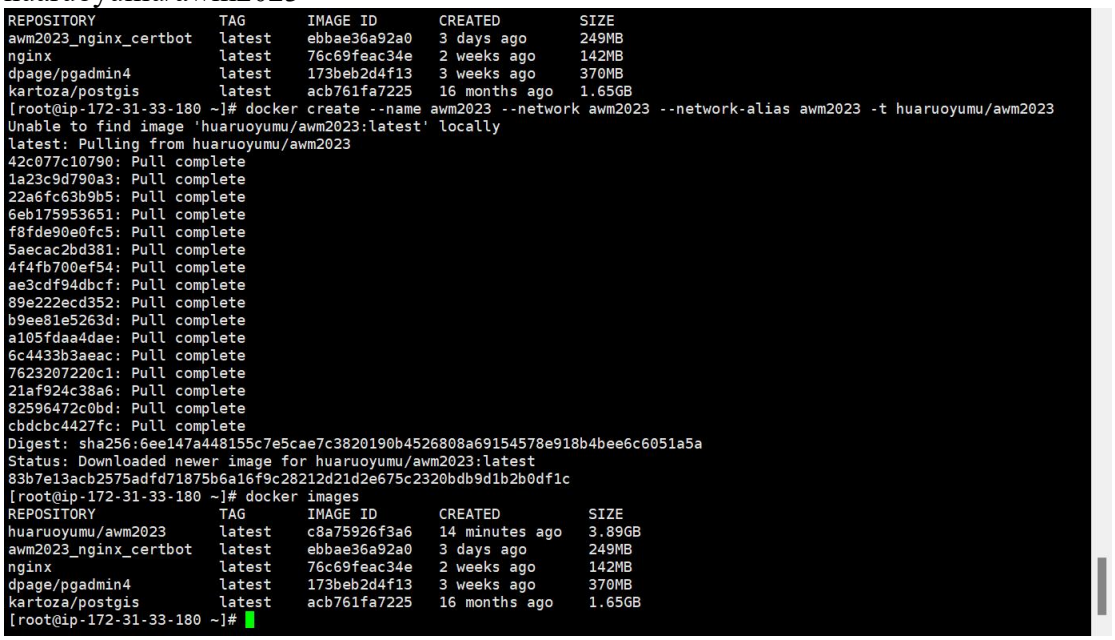
Push the image to the Docker Hub:
First, log in the docker account on the image machine



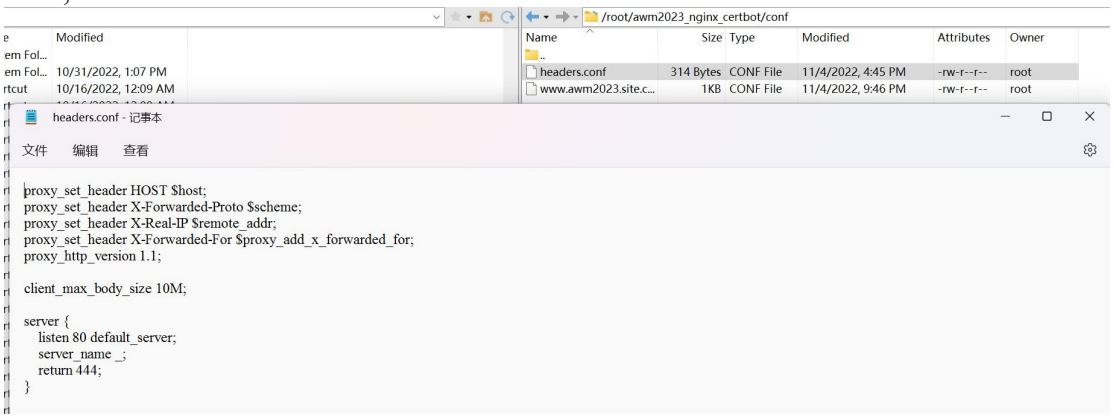
Second, push the image to the Docker Hub by using:
docker push huaruoyumu/awm2023:latest



9. Get the pushed image to cloud server and create the container.
Use: docker create --name awm2023 --network awm2023 --network-alias awm2023 -t huaruoyumu/awm2023



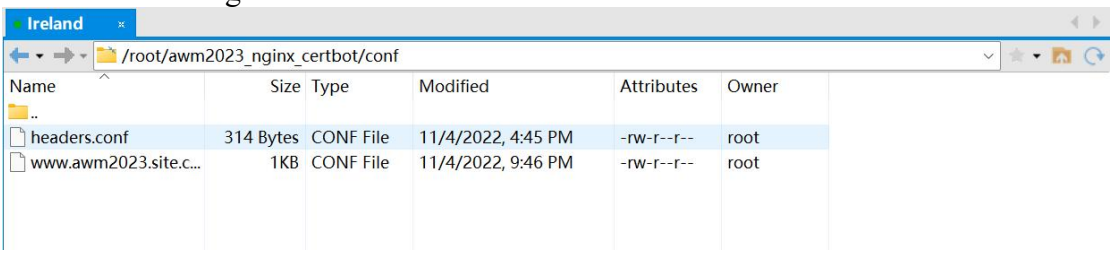
10. Config the Nginx.
First, create the headers.conf.



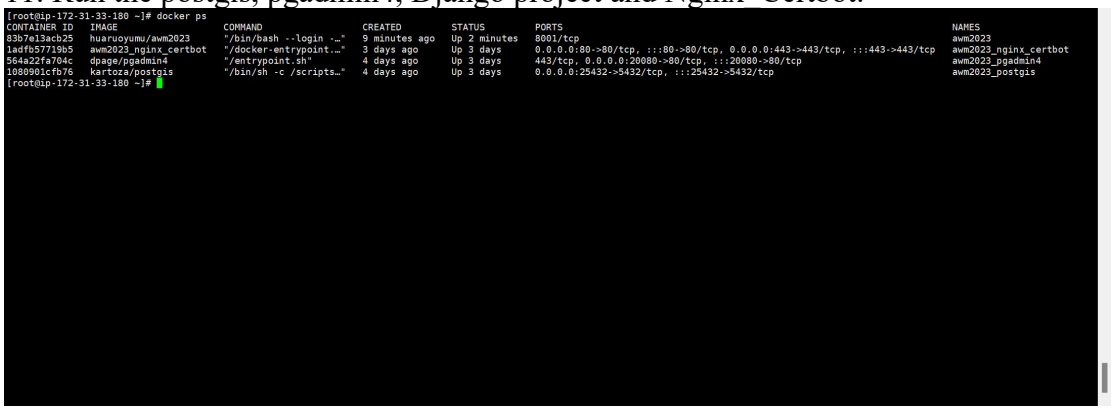
Second, create www.awm2023.site.conf.

```
server {
    listen 80;
    server_name www.awm2023.site;

        location / {
            return 301 https://$host$request_uri;
        }

        location /.well-known/acme-challenge/ {
            root /var/www/certbot;
        }
    }

server {
    listen 443 ssl;

    root /usr/share/nginx/html;
    index index.html;

    server_name www.awm2023.site;

    ssl_certificate /etc/letsencrypt/live/awm2023.site/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/awm2023.site/privkey.pem;

    #include /etc/letsencrypt/options-ssl-nginx.conf;
    #ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

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

    location /pgadmin4 {
        proxy_set_header X-Script-Name /pgadmin4;
        proxy_pass http://awm2023_pgadmin4;
    }

    # Some updates here
    # include uwsgi_params;
    location / {
        proxy_pass http://awm2023:8001;
    }

    # uwsgi_param Host $host;
    # uwsgi_param X-Real-IP $remote_addr;
    # uwsgi_param X-Forwarded-For $proxy_add_x_forwarded_for;
    # uwsgi_param X-Forwarded-Proto $http_x_forwarded_proto;
```

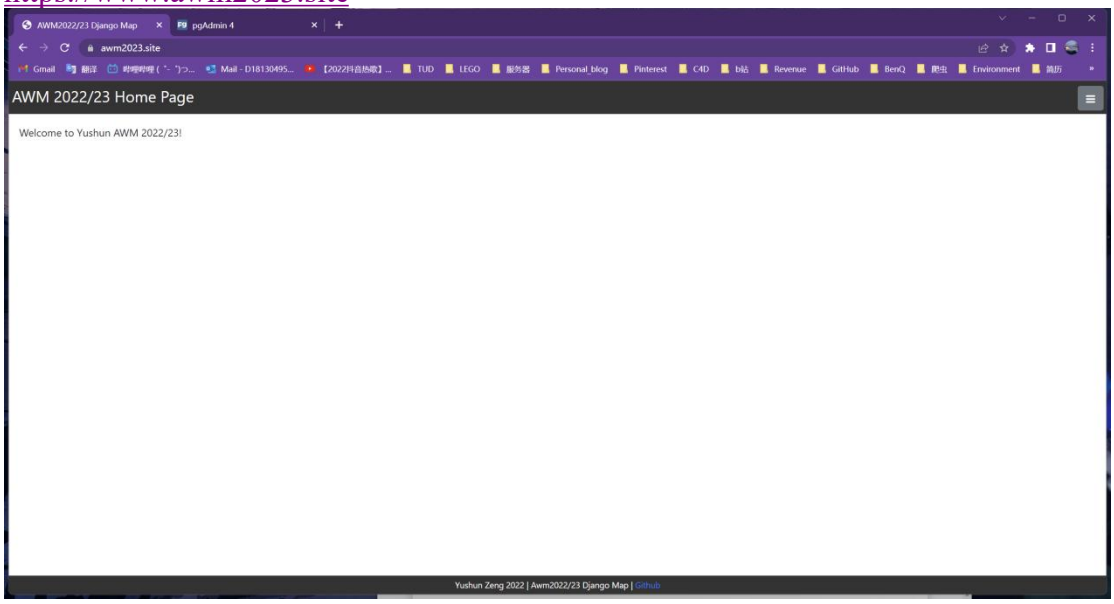
Structure of the nginx conf folder.



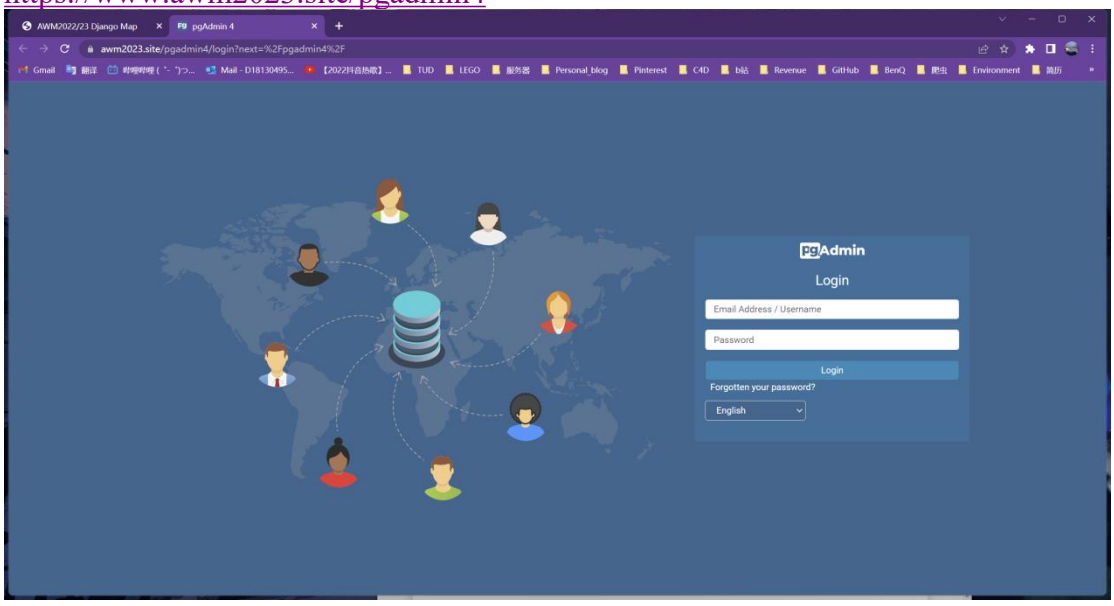
11. Run the postgis, pgadmin4, Django project and Nginx Certbot.



12. Visit https://www.awm2023.site and https://www.awm2023.site/pgadmin4
https://www.awm2023.site



https://www.awm2023.site/pgadmin4



Connected to the postgis