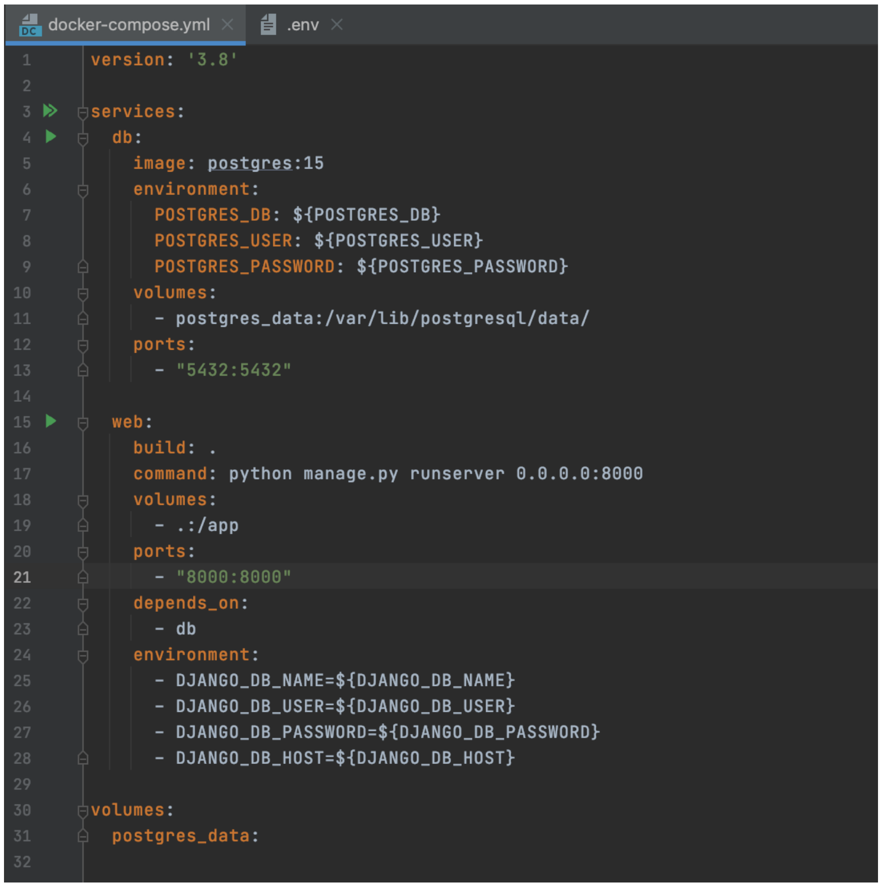
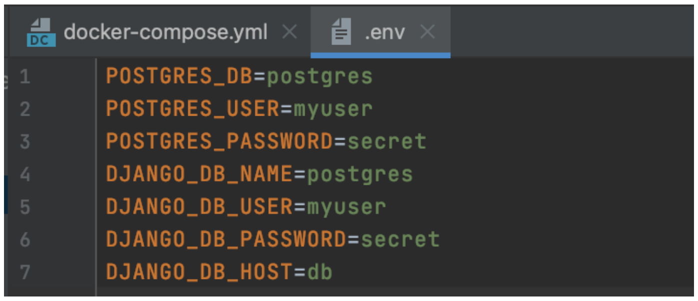
Sarsengaliyev Zhaisan, ID: 23MD0430

Assignment 2, Web app dev

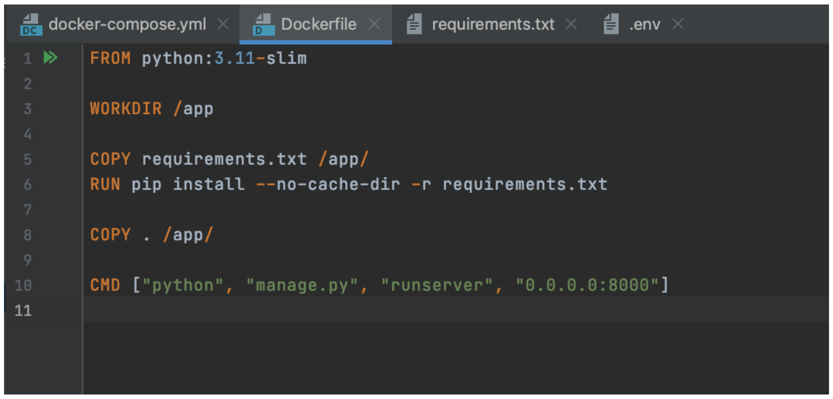
1. Docker Compose



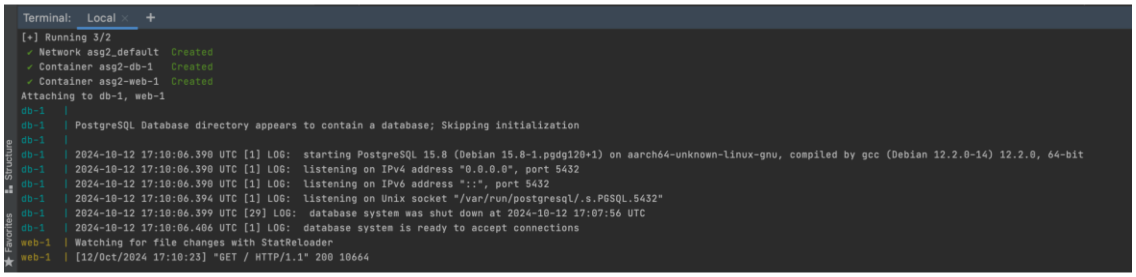
```yaml
version: '3.8'

services:
  db:
    image: postgres:15
    environment:
      POSTGRES_DB: ${POSTGRES_DB}
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data/
    ports:
      - "5432:5432"

  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/app
    ports:
      - "8000:8000"
    depends_on:
      - db
    environment:
      - DJANGO_DB_NAME=${DJANGO_DB_NAME}
      - DJANGO_DB_USER=${DJANGO_DB_USER}
      - DJANGO_DB_PASSWORD=${DJANGO_DB_PASSWORD}
      - DJANGO_DB_HOST=${DJANGO_DB_HOST}

volumes:
  postgres_data:

```



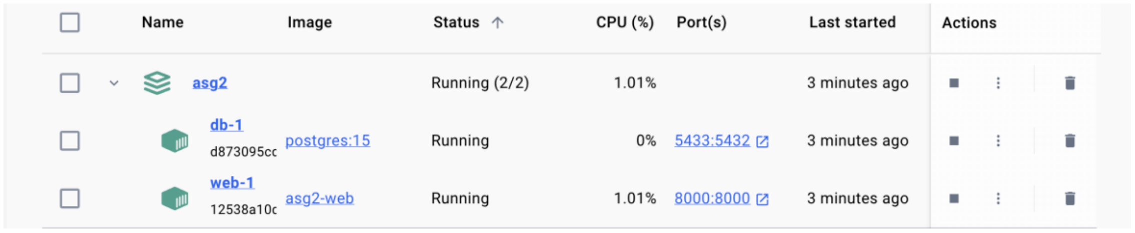
```
POSTGRES_DB=postgres
POSTGRES_USER=myuser
POSTGRES_PASSWORD=secret
DJANGO_DB_NAME=postgres
DJANGO_DB_USER=myuser
DJANGO_DB_PASSWORD=secret
DJANGO_DB_HOST=db
```



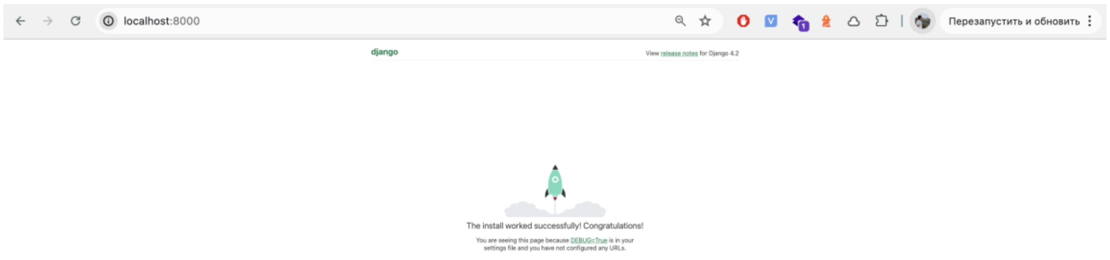
```
FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt /app/
RUN pip install --no-cache-dir -r requirements.txt

COPY . /app/

CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```



```
[+] Running 3/2
 ✓ Network asg2_default  Created
 ✓ Container asg2-db-1   Created
 ✓ Container asg2-web-1  Created
Attaching to db-1, web-1
db-1   |
db-1   | PostgreSQL Database directory appears to contain a database; Skipping initialization
db-1   |
db-1   | 2024-10-12 17:10:06.390 UTC [1] LOG:  starting PostgreSQL 15.8 (Debian 15.8-1.pgdg120+1) on aarch64-unknown-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
db-1   | 2024-10-12 17:10:06.390 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
db-1   | 2024-10-12 17:10:06.390 UTC [1] LOG:  listening on IPv6 address "::", port 5432
db-1   | 2024-10-12 17:10:06.394 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db-1   | 2024-10-12 17:10:06.399 UTC [29] LOG:  database system was shut down at 2024-10-12 17:07:56 UTC
db-1   | 2024-10-12 17:10:06.406 UTC [1] LOG:  database system is ready to accept connections
web-1  | Watching for file changes with StatReloader
web-1  | [12/Oct/2024 17:10:23] "GET / HTTP/1.1" 200 10664
```



Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
asg2		Running (2/2)	1.01%		3 minutes ago	
db-1 d873095cc	postgres:15	Running	0%	5433:5432	3 minutes ago	
web-1 12538a10c	asg2-web	Running	1.01%	8000:8000	3 minutes ago	



View release notes for Django 4.2

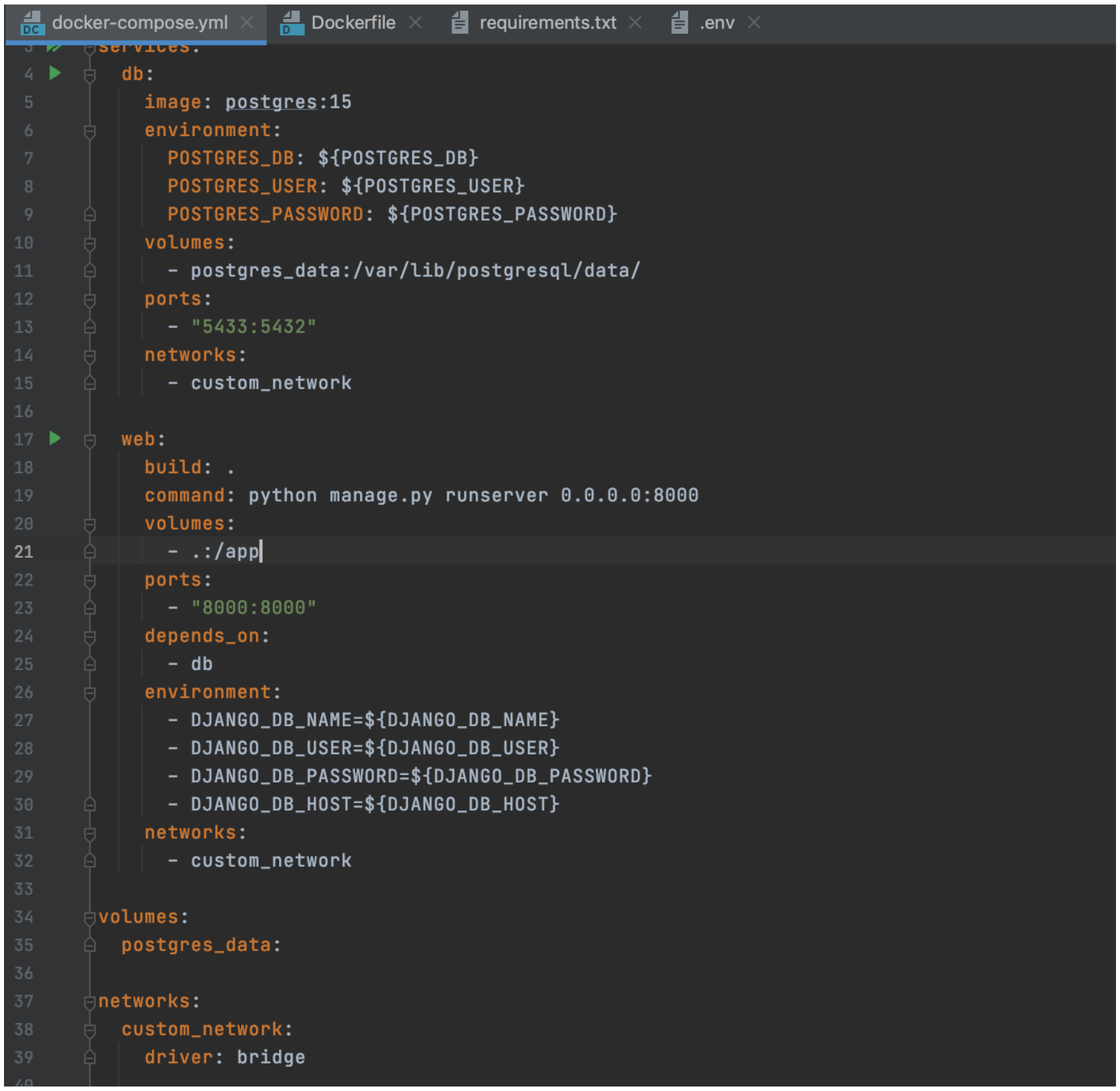
The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

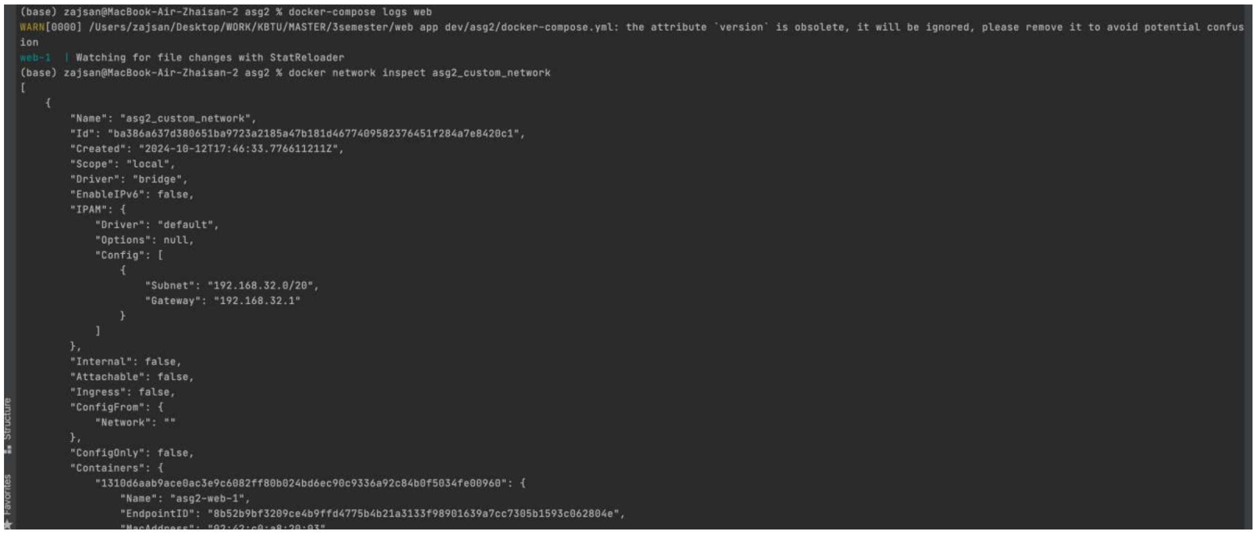
We used 3.8 version. DB service is based on PostgreSQL image 15.
The web service builds Django app from local directory, the local directory is mounted into container app. Port 8000 is exposed making Django app accessible at localhost:8000.
The ‘depends_on’ ensures that db service starts before web service
Docker volume ‘postgres_data’ is used to persist postgre data. This ensures that data will not be lost when container will be stopped.

2. Docker Networking and Volumes

We configured a custom docker network in compose file to allow communication between Django app and db. We verified it by inspecting the custom network by ‘docker network inspect asg2_custom_network’.

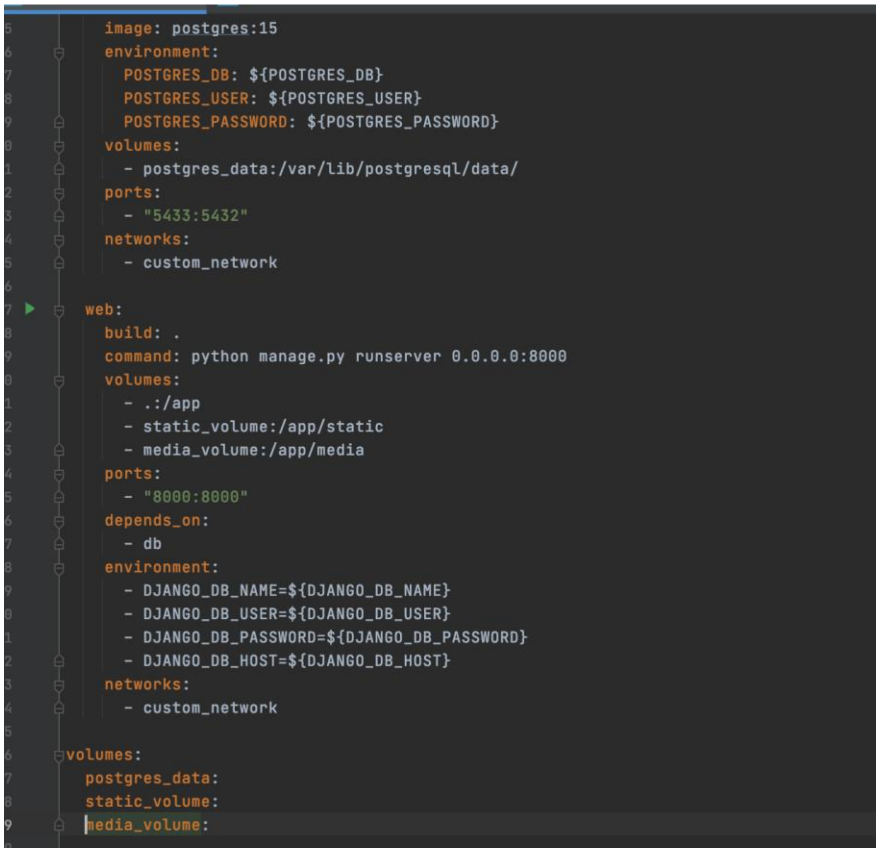


```yaml
services:
  db:
    image: postgres:15
    environment:
      POSTGRES_DB: ${POSTGRES_DB}
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data/
    ports:
      - "5433:5432"
    networks:
      - custom_network

  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/app
    ports:
      - "8000:8000"
    depends_on:
      - db
    environment:
      - DJANGO_DB_NAME=${DJANGO_DB_NAME}
      - DJANGO_DB_USER=${DJANGO_DB_USER}
      - DJANGO_DB_PASSWORD=${DJANGO_DB_PASSWORD}
      - DJANGO_DB_HOST=${DJANGO_DB_HOST}
    networks:
      - custom_network

volumes:
  postgres_data:

networks:
  custom_network:
    driver: bridge
```



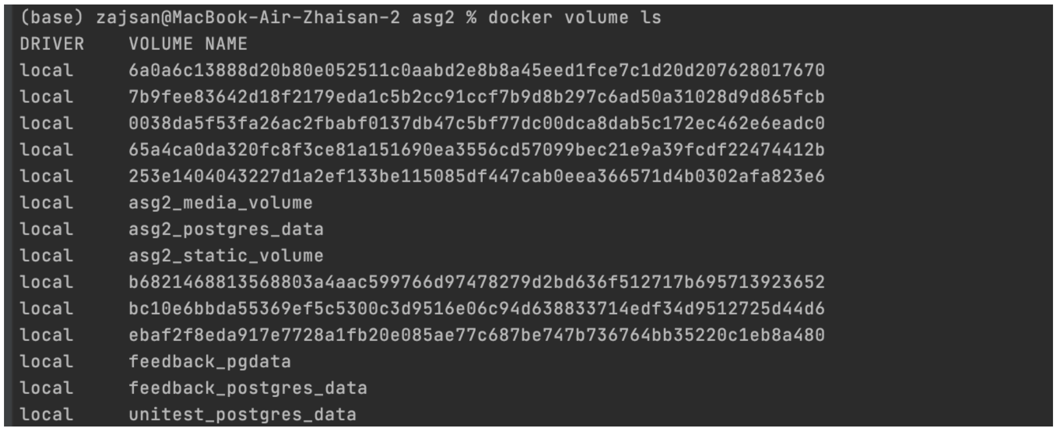
```
(base) zajsan@MacBook-Air-Zhaisan-2 asg2 % docker-compose logs web
WARN[0000] /Users/zajsan/Desktop/WORK/KBTU/MASTER/3semester/web app dev/asg2/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confus
ion
web-1  | Watching for file changes with StatReloader
(base) zajsan@MacBook-Air-Zhaisan-2 asg2 % docker network inspect asg2_custom_network
[
    {
        "Name": "asg2_custom_network",
        "Id": "ba386a637d380651ba9723a2185a47b181d4677409582376451f284a7e8420c1",
        "Created": "2024-10-12T17:46:33.776611211Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "192.168.32.0/20",
                    "Gateway": "192.168.32.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "1310d6aab9ace0ac3e9c6082ff80b024bd6ec90c9336a92c84b0f5034fe00960": {
                "Name": "asg2-web-1",
                "EndpointID": "8b52b9bf3209ce4b9ffd4775b4b21a3133f98901639a7cc7305b1593c062804e",
                "MacAddress": "02:42:c0:a8:20:03"
```

We added volumes to persist statis and uploaded media files



```yaml
    image: postgres:15
    environment:
      POSTGRES_DB: ${POSTGRES_DB}
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data/
    ports:
      - "5433:5432"
    networks:
      - custom_network

  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/app
      - static_volume:/app/static
      - media_volume:/app/media
    ports:
      - "8000:8000"
    depends_on:
      - db
    environment:
      - DJANGO_DB_NAME=${DJANGO_DB_NAME}
      - DJANGO_DB_USER=${DJANGO_DB_USER}
      - DJANGO_DB_PASSWORD=${DJANGO_DB_PASSWORD}
      - DJANGO_DB_HOST=${DJANGO_DB_HOST}
    networks:
      - custom_network

volumes:
  postgres_data:
  static_volume:
  media_volume:
```

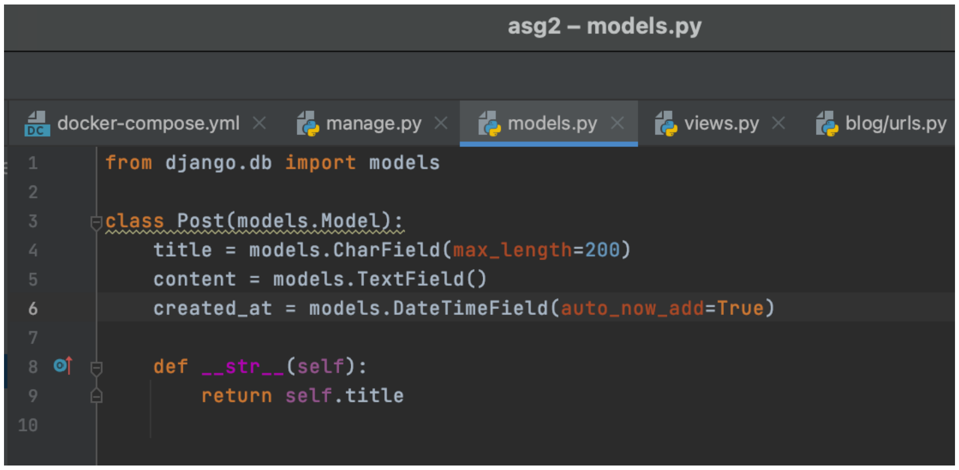
We can verify it by ‘docker volume ls’, so we see asg2_media volume, asg2_postgres_data, asg2_static_volume.



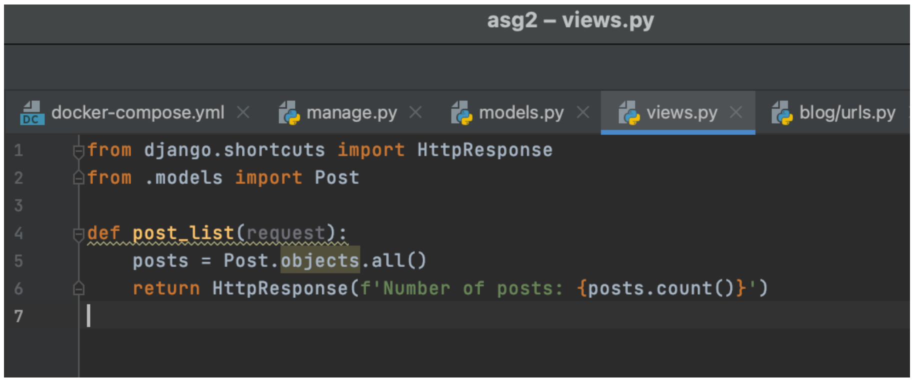
```
(base) zajsan@MacBook-Air-Zhaisan-2 asg2 % docker volume ls
DRIVER    VOLUME NAME
local     6a0a6c13888d20b80e052511c0aabd2e8b8a45eed1fce7c1d20d207628017670
local     7b9fee83642d18f2179eda1c5b2cc91ccf7b9d8b297c6ad50a31028d9d865fcb
local     0038da5f53fa26ac2fbabf0137db47c5bf77dc00dca8dab5c172ec462e6eadc0
local     65a4ca0da320fc8f3ce81a151690ea3556cd57099bec21e9a39fcdf22474412b
local     253e1404043227d1a2ef133be115085df447cab0eea366571d4b0302afa823e6
local     asg2_media_volume
local     asg2_postgres_data
local     asg2_static_volume
local     b6821468813568803a4aac599766d97478279d2bd636f512717b695713923652
local     bc10e6bbda55369ef5c5300c3d9516e06c94d638833714edf34d9512725d44d6
local     ebaf2f8eda917e7728a1fb20e085ae77c687be747b736764bb35220c1eb8a480
local     feedback_pgdata
local     feedback_postgres_data
local     unitest_postgres_data
```

The custom_network ensures that services are isolated from external systems. This provides security. Our new volumes are used to persist statis files (like css, javascript, images) and user uploaded media files. This ensures that they will not be lost when django container is rebuilt.

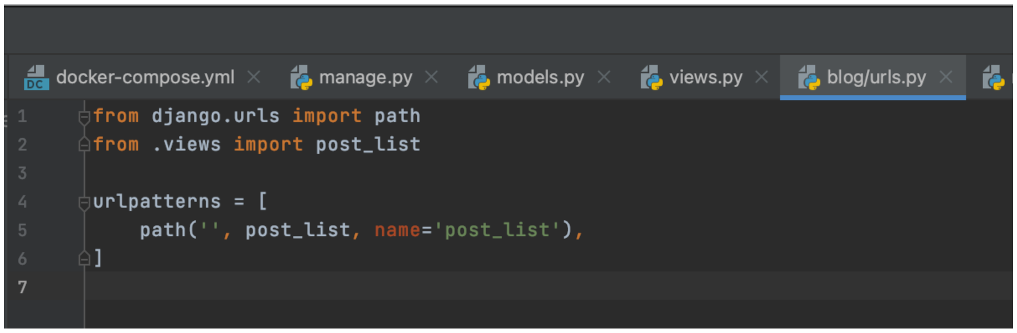
3. Django Application Setup



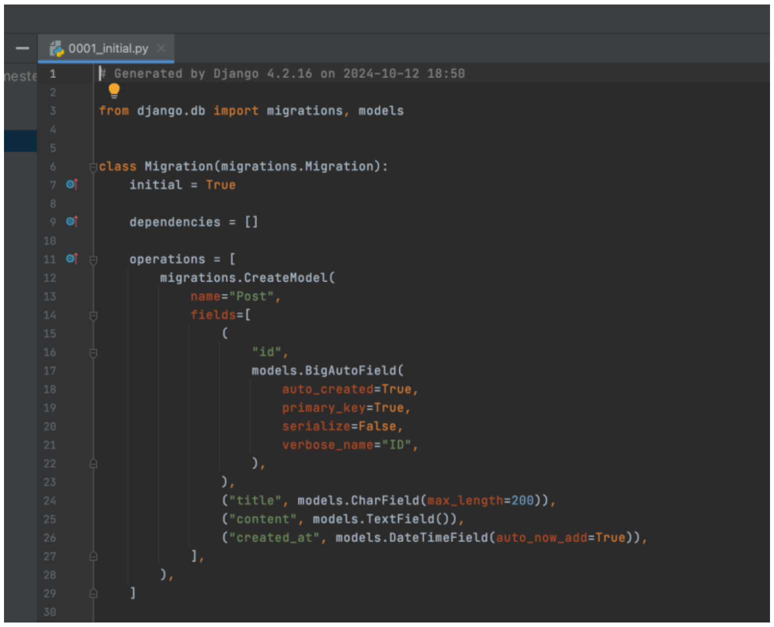
```python
from django.db import models

class Post(models.Model):
    title = models.CharField(max_length=200)
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.title

```



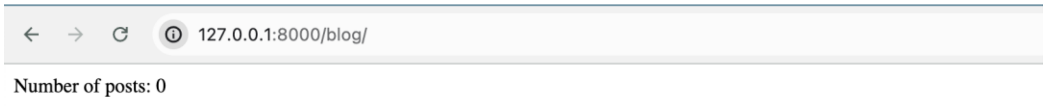
```python
from django.shortcuts import HttpResponse
from .models import Post

def post_list(request):
    posts = Post.objects.all()
    return HttpResponse(f'Number of posts: {posts.count()}')

```



```python
from django.urls import path
from .views import post_list

urlpatterns = [
    path('', post_list, name='post_list'),
]
```



```python
# Generated by Django 4.2.16 on 2024-10-12 18:50

from django.db import migrations, models


class Migration(migrations.Migration):
    initial = True

    dependencies = []

    operations = [
        migrations.CreateModel(
            name="Post",
            fields=[
                (
                    "id",
                    models.BigAutoField(
                        auto_created=True,
                        primary_key=True,
                        serialize=False,
                        verbose_name="ID",
                    ),
                ),
                ("title", models.CharField(max_length=200)),
                ("content", models.TextField()),
                ("created_at", models.DateTimeField(auto_now_add=True)),
            ],
        ),
    ]
```



```
Number of posts: 0
```

So, we created basic model which has fields like title, content, created_at. A simple view 'post_list' was created to display the number of blog posts. The URLs for the blog app were set up in blog/urls.py and included in the project's main urls.py. We also set up database schema, this created the required tables in postgresql database for Django.

Conclusion

We set up Django project with PostgreSQL using Docker, created a basic app, and applied database migrations. Docker helped make everything run smoothly by keeping the setup consistent and easy to manage. This way the project can work the same on any computer or server.