Report

Title of the Assignment: Assignment 2, Web app dev

Student's Name: Abylay Aitbanov 23MD0377

Date of Submission: 13.10.2024

Table of Contents

Introduction

The goal of this assignment is to gain hands-on experience with Django and Docker, focusing on Docker Compose, Docker networking, and volumes. Students will set up a Django application within a Docker environment and document the process.

Docker Compose

Configuration

docker-compose.yml × manage.py Dockerfile

```yaml
version: '3'

services:
  db:
    image: postgres:13
    environment:
      POSTGRES_DB: mydatabase
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: mypassword
    volumes:
      - postgres_data:/var/lib/postgresql/data
    networks:
      - django-network

  web:
    build: .
    command: python3 manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/app
      - static_volume:/app/static
    ports:
      - "8000:8000"
    depends_on:
      - db
    environment:
      - DB_NAME=mydatabase
      - DB_USER=myuser
      - DB_PASSWORD=mypassword
      - DB_HOST=db
    networks:
      - django-network

volumes:
  postgres_data:
  static_volume:

networks:
  django-network:
```

In this assignment, it includes two main services: db for PostgreSQL and web for the Django application. The configuration specifies environment variables, network settings, and volume mappings to ensure data persistence and proper communication between services.

Breakdown of the Configuration

db:
image: postgres:13 - > it uses the official PostgreSQL image with version 13

environment:
POSTGRES_DB: mydatabase
POSTGRES_USER: myuser

POSTGRES_PASSWORD: mypassword

POSTGRES_DB: The name of the database that will be created when the PostgreSQL container is initialized.
POSTGRES_USER: The username for connecting to the PostgreSQL database.
POSTGRES_PASSWORD: The password for the specified user. This is crucial for database authentication.

volumes:
- postgres_data:/var/lib/postgresql/data -> Maps a named volume (postgres_data) to the PostgreSQL data directory inside the container

networks:
- django-network - > Connects the db service to a custom network named django-network

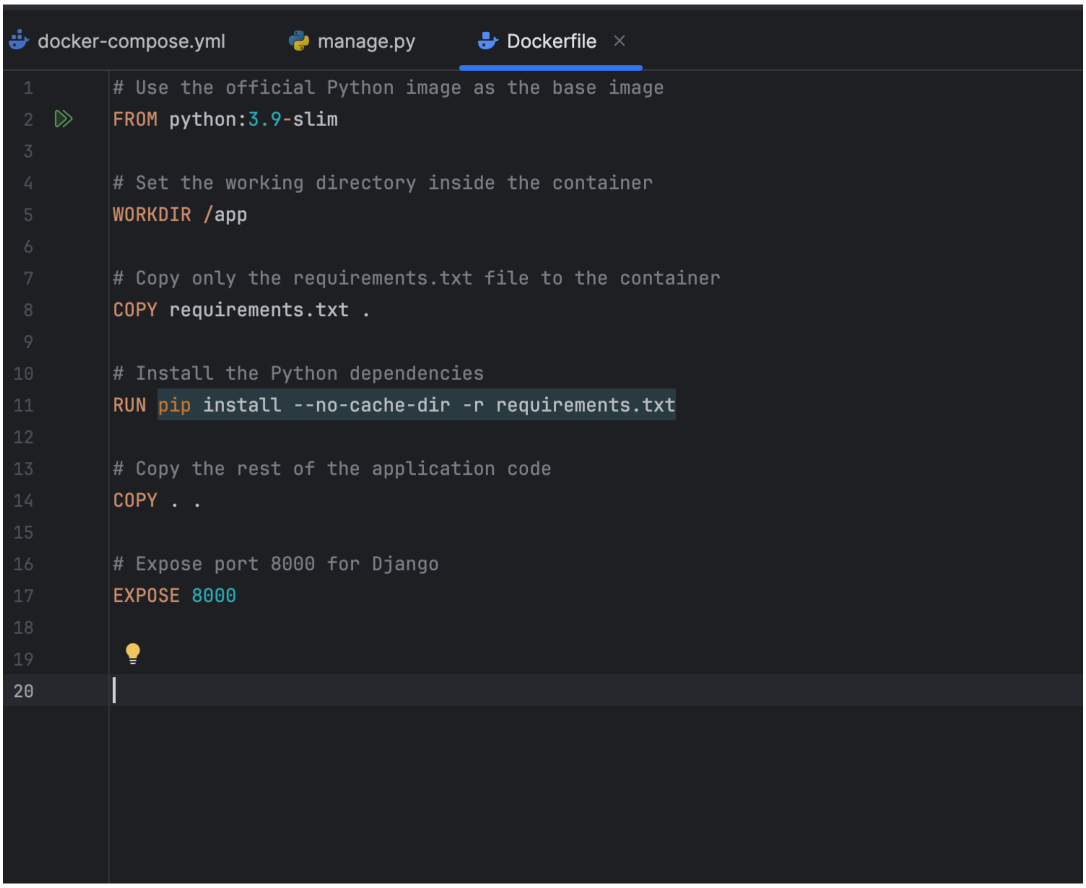
web:
build: . -> Docker image for this service will be built from the current directory

command: python3 manage.py runserver 0.0.0.0:8000 -> command to run when the container starts

volumes:
- .:/app -> mounts the current directory
- static_volume:/app/static -> maps a named volume to the static files

ports:
- "8000:8000" -> port 8000 on the host machine, mapping it to port 8000 on the container

Configuration of Dockerfile



```dockerfile
# Use the official Python image as the base image
FROM python:3.9-slim

# Set the working directory inside the container
WORKDIR /app

# Copy only the requirements.txt file to the container
COPY requirements.txt .

# Install the Python dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application code
COPY . .

# Expose port 8000 for Django
EXPOSE 8000


```

FROM python:3.9-slim -> official Python image version 3.9, with a "slim" variant that is smaller in size
WORKDIR /app -> This directive sets the working directory inside the container to /app
COPY requirements.txt . -> This command copies files from the host machine into the Docker image
RUN pip install --no-cache-dir -r requirements.txt -> to install the dependencies listed in requirements.txt. The --no-cache-dir option is used to prevent pip from caching
COPY . . -> This command copies all files and directories from the current directory on the host machine into the current working directory
EXPOSE 8000 -> This is the port used by the Django development server

Build and Run

```
docker-compose build

[+] Building 3.0s (11/11) FINISHED                                                                                                                                           docker:desktop-linux
 => [web internal] load build definition from Dockerfile                                                                                                                                     0.0s
 => => transferring dockerfile: 412B                                                                                                                                                         0.0s
 => [web internal] load .dockerignore                                                                                                                                                        0.0s
 => => transferring context: 63B                                                                                                                                                             0.0s
 => [web internal] load metadata for docker.io/library/python:3.9-slim                                                                                                                       2.9s
 => [web auth] library/python:pull token for registry-1.docker.io                                                                                                                            0.0s
 => [web 1/5] FROM docker.io/library/python:3.9-slim@sha256:49f94609e5a997dc16086a66ac9664591854031d48e375945a9dbf4d1d53abbc                                                                 0.0s
 => [web internal] load build context                                                                                                                                                        0.0s
 => => transferring context: 7.49kB                                                                                                                                                          0.0s
 => CACHED [web 2/5] WORKDIR /app                                                                                                                                                            0.0s
 => CACHED [web 3/5] COPY requirements.txt .                                                                                                                                                 0.0s
 => CACHED [web 4/5] RUN pip3 install -r requirements.txt                                                                                                                                    0.0s
 => [web 5/5] COPY . .                                                                                                                                                                       0.0s
 => [web] exporting to image                                                                                                                                                                 0.0s
 => => exporting layers                                                                                                                                                                      0.0s
 => => writing image sha256:b27f7773a866bae9d6b9883aa139c593e5f1fce1381c2cd0efc88e1b4d798a1b                                                                                                 0.0s
 => => naming to docker.io/library/ass2-web                                                                                                                                                  0.0s
```

```
(venv) ~/Desktop/KBTU/Web-app-dev-2024/Ass2 git:[main]
docker-compose up -d

[+] Running 3/3
 ✔ Network ass2_django-network  Created                                    0.0s
 ✔ Container ass2-db-1          Started                                    0.0s
 ✔ Container ass2-web-1         Started                                    0.0s
```

ass2			Running (2/2)	0%		40 minutes ago
web-1 8c098d2ea549	ass2-web		Running	0%	8000:8000	40 minutes ago
db-1 47e5c97cf549	postgres:13		Running	0%		53 minutes ago

docker-compose up –build -> This command compiles the images defined in the Dockerfile and starts the services in the docker-compose.yml.

Docker Networking and Volumes

Networking

```
networks:
  django-network:
```

Networking allows to communication between containers. In this setup, a network named django-network was created to communicate between the Django web service and the PostgreSQL database service. This setup enhances security and isolates container traffic from the host network.

Volumes

```
    volumes:
      - .:/app
      - static_volume:/app/static
    ports:
```

```
volumes:
  - postgres_data:/var/lib/postgresql/data
```

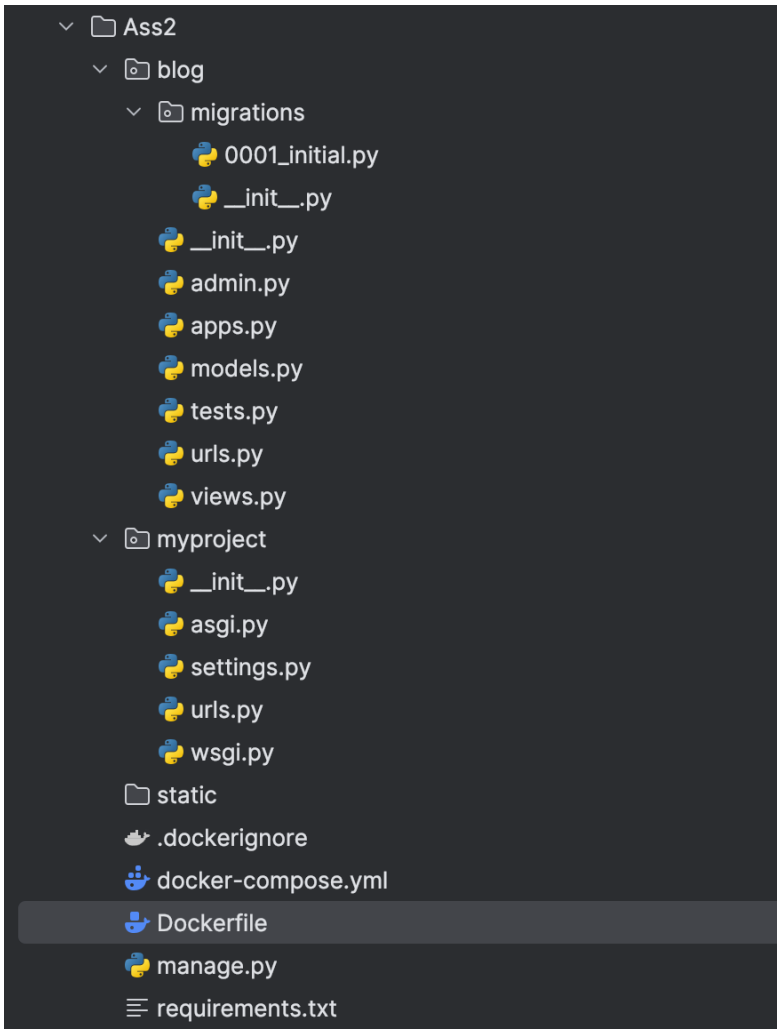
Volumes were implemented to ensure data persistence for the PostgreSQL database. The postgres_data volume maps to the database's data directory, ensuring that data remains intact across container restarts.

Findings

The use of Docker networking and volumes provides significant advantages:
Data Persistence -> volumes ensure that the database retains data
Isolation - > custom networks enhance security by isolating communication between containers

Django Application Setup

Project Structure



Standard structure -> including essential directories for applications, static files, and templates. The main application, blog, contains models, views, and URLs for managing blog posts.

```python
from django.db import models


4 usages  abylayaitbanov
class Post(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField()

    abylayaitbanov
    def __str__(self):
        return self.title
```

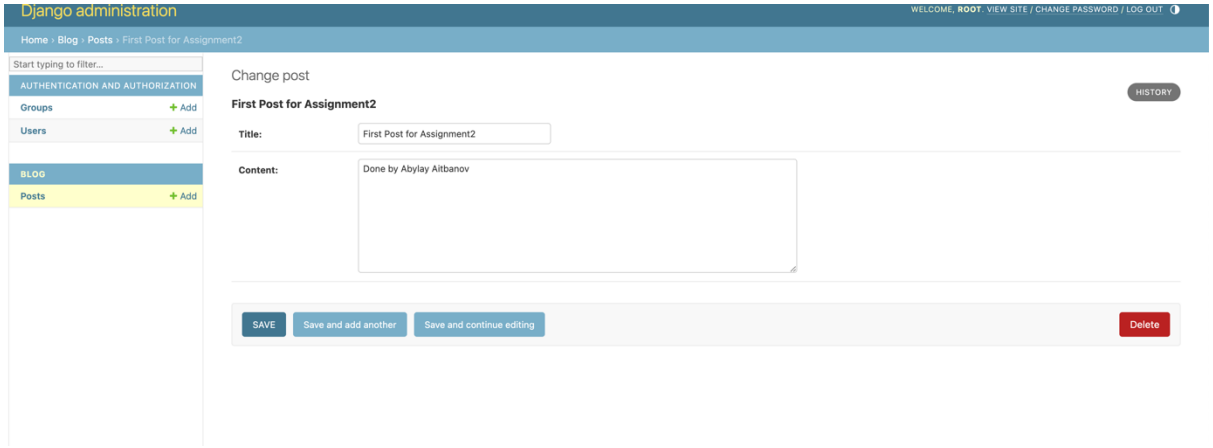
In blog.models we have simple model Post that habe 2 attributes title and content, And resulting self.title

docker-compose.yml models.py admin.py manage.py Dockerfile

```python
from django.contrib import admin
from .models import Post

# Register your models here.
admin.site.register(Post)
```

In blog.admins I have register part, to create some posts in admin panel



```python
# Generated by Django 4.2.16 on 2024-10-13 14:01

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=100)),
                ('content', models.TextField()),
            ],
        ),
    ]
```

Here we have migration files, that shows us the creation of model

```python
from django.http import JsonResponse
from .models import Post


def index(request):
    posts = Post.objects.all()
    posts_data = [{'id': post.id, 'title': post.title} for post in posts]
    return JsonResponse(posts_data, safe=False)  # safe=False allows non-dict objects
```

Here we have blog.views file that take all Post object file from Db and return JsonResponse

```python
"""
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('blog/', include('blog.urls')),
]
```

Main urls.py file where we redirect out requests that we need, here we have admin/, and blog/ urls

docker-compose run web django-admin startproject myproject . -> command to start Django project

docker-compose run web python manage.py startapp blog -> to create app (blog)

Database Configuration

The PostgreSQL database is integrated with Django through the settings defined in settings.py

```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': os.environ.get('DB_NAME'),
        'USER': os.environ.get('DB_USER'),
        'PASSWORD': os.environ.get('DB_PASSWORD'),
        'HOST': os.environ.get('DB_HOST'),  # This should be 'db'
        'PORT': '5432',
    }
}
```

docker-compose.yml × | models.py | tests.py | blog/urls.py | views.py

```yaml
version: '3'

services:
  db:
    image: postgres:13
    environment:
      POSTGRES_DB: mydatabase
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: mypassword
    volumes:
      - postgres_data:/var/lib/postgresql/data
    networks:
      - django-network

  web:
    build: .
```

All the os.environ was define in docker-compose file

Findings

Developing a Django application within Docker has provided numerous benefits, including consistent development environments and ability to easily scale and manage services

Conclusion

This assignment shows that the key learnings from setting up a Django application using Docker. The combination of Docker and Django not only streamlines the development process but also improves application deployment and management.

References

Official Docker docs: https://docs.docker.com/
Official Django docs: https://www.djangoproject.com/
My github page: https://github.com/AA19BD