MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF KAZAKHSTAN

KAZAKH-BRITISH TECHNICAL UNIVERSITY

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

REPORT

Web Application Development

Assignment 2

Presented to Serek A.G.

Done by Kabyl Dauren

Student ID: 23MD0452

Almaty 2024

Content

Introduction

In this work we would show how to work with docker compose (Sections 2 and 3) and how to create simple application on the django framework (Section 4).

Docker compose is a tool that allows developers to easily run containers with many different services inside [1]. This allows developers to more easily and better monitor running containers.

Django is a python framework that makes web development much easier [2]. It has much potential and is easy to understand, so with django you can create different types of applications.

You can see the code by this link:

Link to Github project: https://github.com/diikiin/assignment2

Docker Compose

Configuration

- **Create a Docker Compose File**
  - Create a docker-compose.yml file for your Django application.
  - Include services for:
    - Django web server
    - PostgreSQL database (or another database of your choice)
- **Define Environment Variables**
  - Use environment variables for database configuration (e.g., DB_NAME, DB_USER, DB_PASSWORD).

As shown in Image 1, in the docker-compose.yml file I wrote setup configuration for Postgresql database and my django project. In the db service I configured the database name, user and password for connecting. In the django service I configured environment variables for connecting to postgres database. Django service depends on this database so if it is not running this service will not run either.

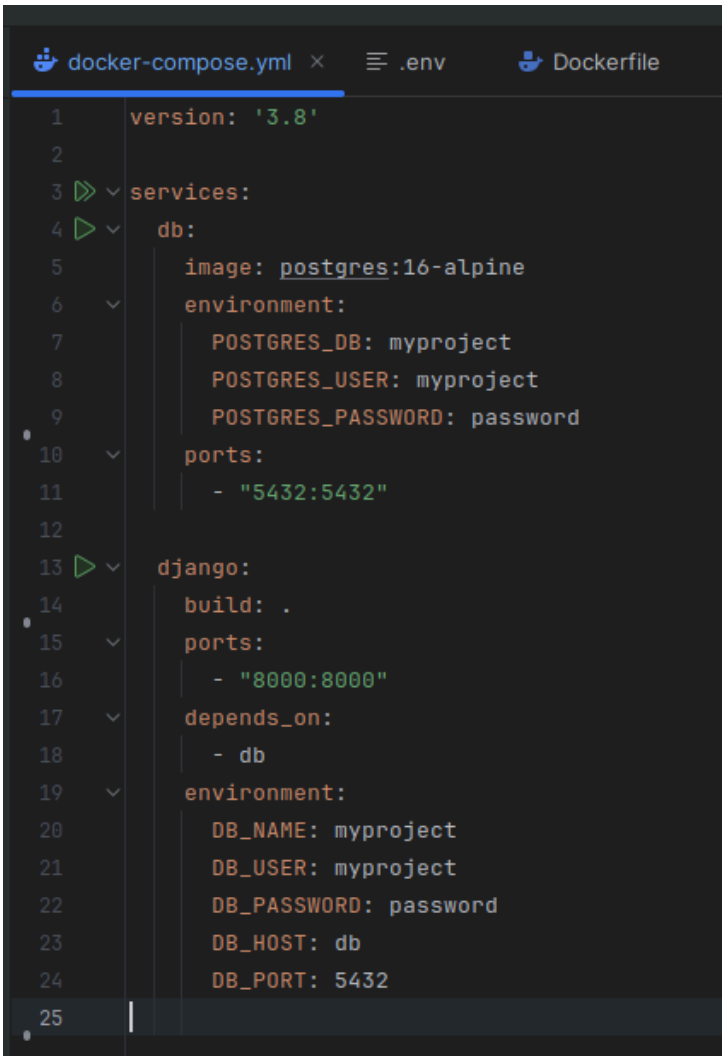

Image 1. docker-compose.yml file configuration

Build and Run

- **Build and Run the Containers**
  - Use docker-compose up to build and run the application.
  - Ensure that the services are running correctly.

With command **docker compose up -d –build** to build and run services in detach mode I started this services. At first it pulled the required image for postgres and built a local Dockerfile image with a python image. Then it installed required python packages and runned django service on port 8000. That they are correctly running you can see in Image 4.

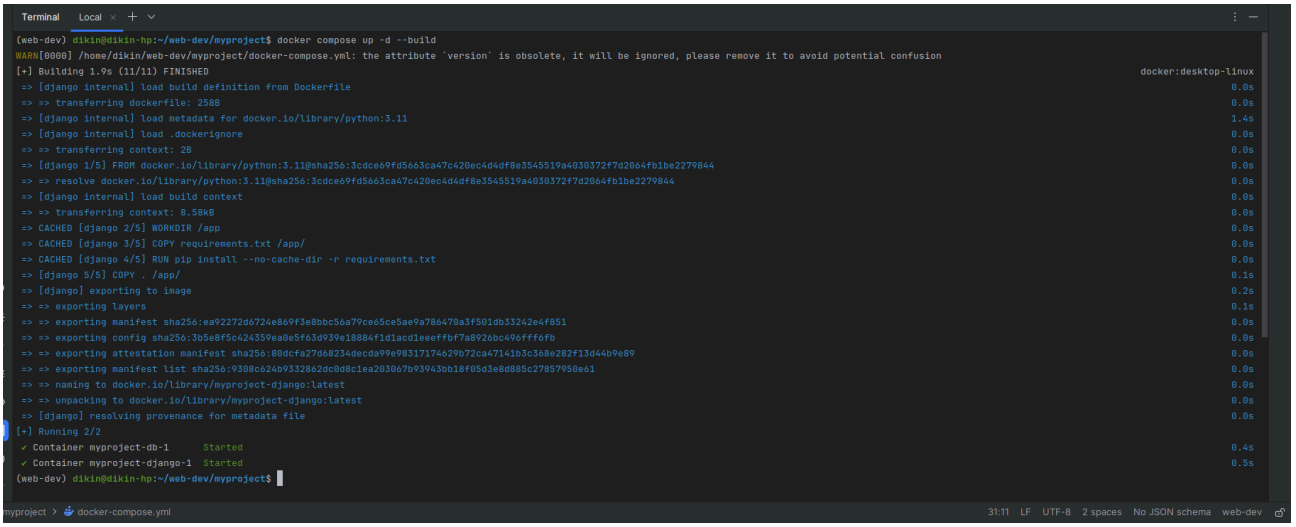

Image 2. Building and Running docker-compose.yml

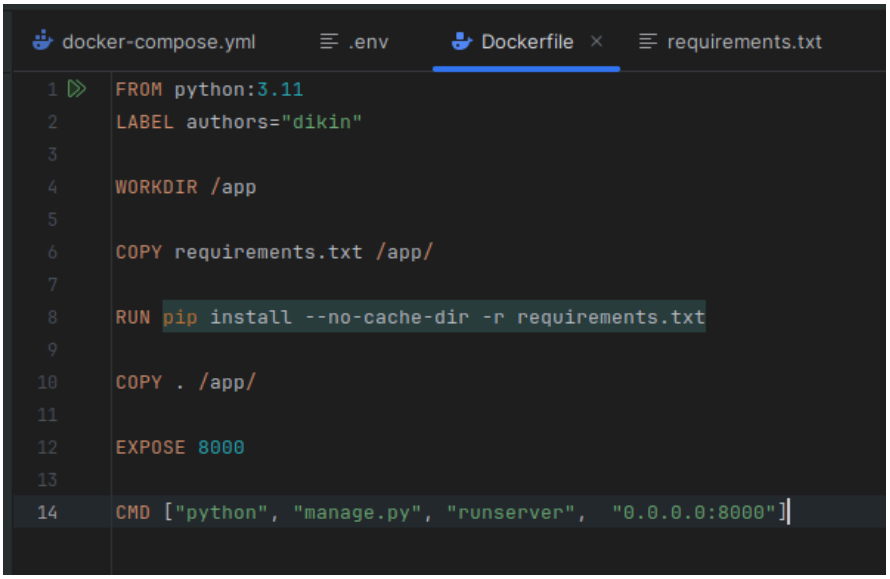

Image 3. Dockerfile

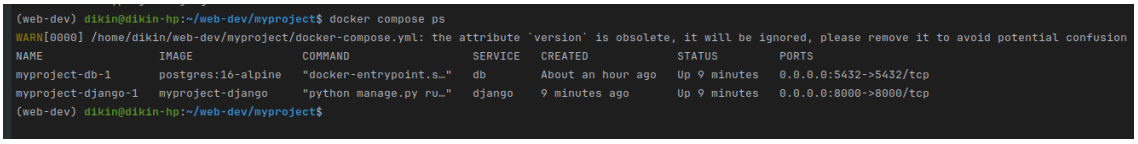

Image 4. Running docker images

Docker Networking and Volumes

Networking

- **Set Up Docker Networking**
  - Define a custom network in your docker-compose.yml file to allow communication between services.
  - Verify that the Django app can connect to the database using the network.

As shown in Image 5, I added network configurations to db. Network **django_network** allows django service to identify the db service host by hostname **db** except IP address of host. So, I can connect to the database like **db:5432** as the address.

Volumes

- **Implement Docker Volumes**
  - Configure a volume in the `docker-compose.yml` file to persist PostgreSQL data.
  - Add a volume for Django to persist uploaded files and static files.

As shown in Image 5, I added volumes configuration for postgres data, static files and uploaded files to db. Data saved to these volumes will be saved even if the container restarts.

Findings

Networks in docker make it easier to interact between services and volumes that work as storages to all data. They make it easier to understand docker structure and working with real apps.

Image 5. Volumes and Networks

Django Application Setup

Project Structure

- **Create a Django Project**
  - Inside the Django service container, create a new Django project using the command django-admin startproject myproject.
  - Create a simple app (e.g., blog) with at least one model and a corresponding view.

As shown in Image 6 and Image 7 I created a blog app in my django project. This app includes a model Post that saves blog posts of blog in database and posts.html view that shows these blogs.

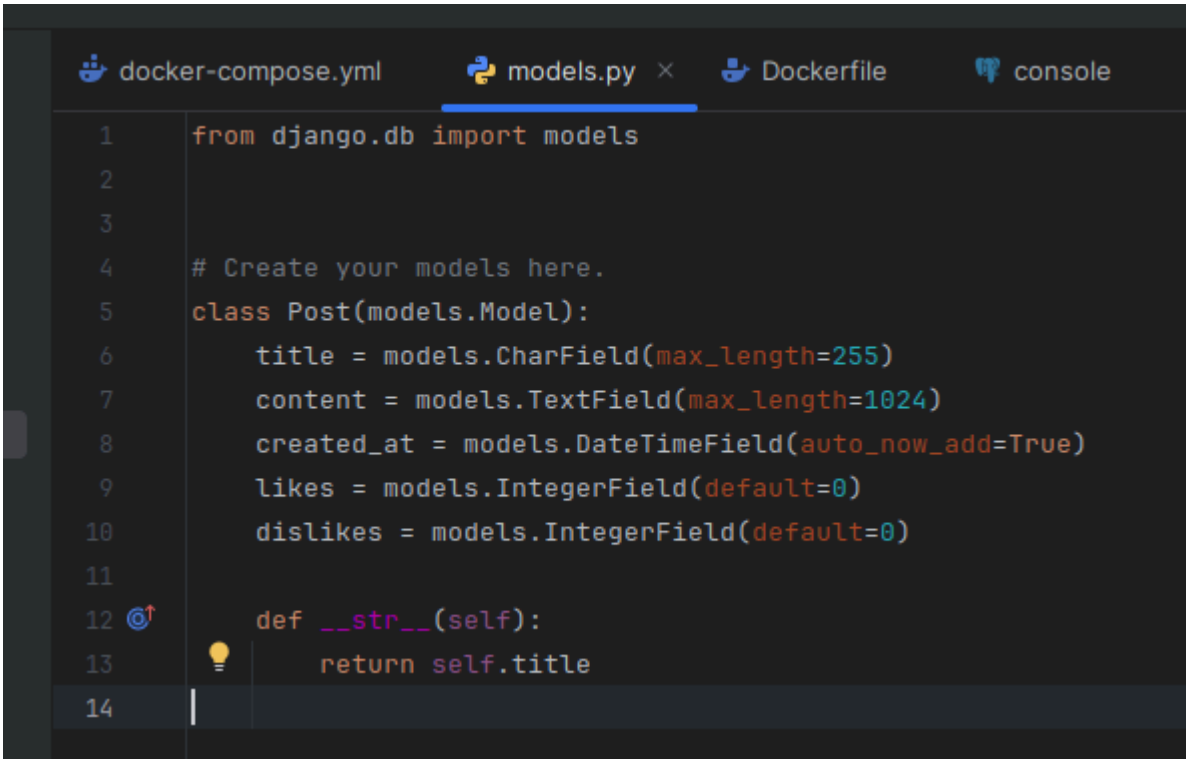

Image 6. Posts model

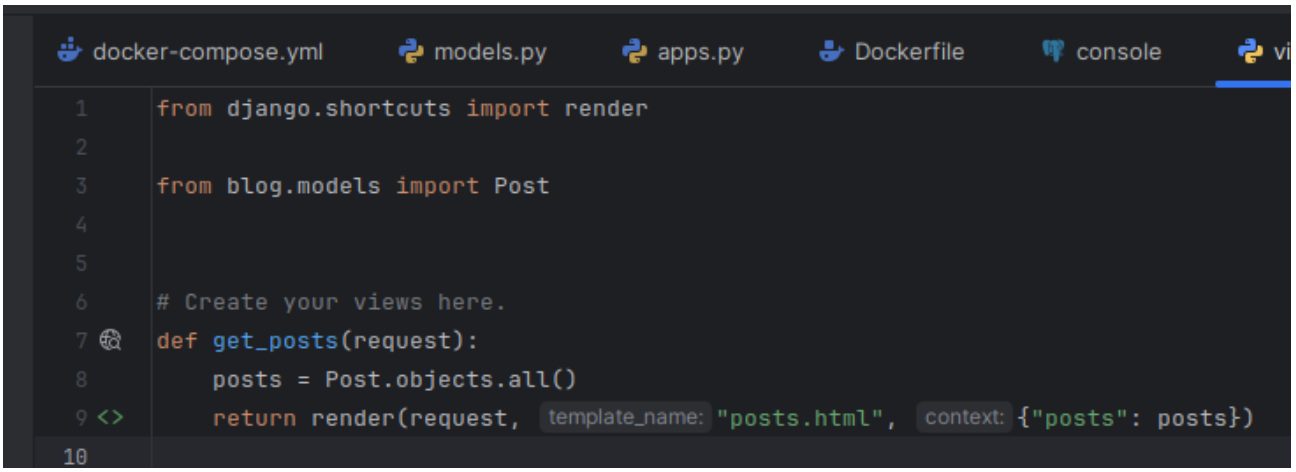

Image 7. Posts view

Database Configuration

- **Configure the Database**
  - Update the Django settings to use the PostgreSQL database configured in your Docker Compose setup.
  - Run migrations to set up the database schema.

As shown in Image 8 I migrated my Posts to db with **python manage.py makemigrations** and **python manage.py migrate** commands.

```
Operations to perform:
  Apply all migrations: admin, auth, blog, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying blog.0001_initial... OK
  Applying sessions.0001_initial... OK
```

Image 8. Database Migrations

Findings

In Django we can easily create an app with endpoints. It uses MVT (Model-View-Template) architecture to easily show models in views. Django ORM automatically creates migration classes according to our models, which makes working with database much easier.

Conclusion

In this work we discussed Docker Compose and Django. We saw how to work with docker containers using Docker Compose and how to create docker networks and volumes. Also, we created a simple app blog that shows blog posts with Django and created migrations using Django Models and Django ORM. It was the first step to developing a fully functional web application using Django, so our future work will be targeted to learn Django much better.

References

1. Docker documentation: https://docs.docker.com/manuals/

2. PostgreSql documentation: https://www.postgresql.org/docs/

3. Django documentation: https://docs.djangoproject.com/en/5.1/

4. Github project: https://github.com/diikiin/assignment2