Web development

Assignment 2: Exploring Django with Docker

Bitanov Assanali (23MD0392)

13.10.2024

Table of Contents

[INTRODUCTION 3](#_Toc179731630)

[Docker Compose 4](#_Toc179731631)

[Docker Networking and Volumes 7](#_Toc179731632)

[Django Application Setup 8](#_Toc179731633)

[Conclusion 10](#_Toc179731634)

[References 11](#_Toc179731635)

# INTRODUCTION

In this assignment, we are going to learn how to work with docker-compose, docker networking, and volumes. 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

First, we create a new directory for the project. Assignment2 is the name of the main directoty. In this folder we create docker-compose.yml. Docker compose is a tool that allows us to create a multi-layered app by defining the paratemers in the docker-compose.yml file. The file is shown in Figure 1.



Figure 1. The docker-compose.yml file. The web part.

We use Dockerfile to build the image that will run the server by using python manage.py command. Also, we define the DB variables to connect to the database. The db part of the docker-compose is shown in Figure 2.



Figure 2. The docker-compose.yml file. The db part.

We use the last version of MySQL and defining the MYSQL variables. The ports in the configuration indicate that port 3306 (the standard MySQL port) inside the container will be available on port 3307 on the local machine.

Now we use docker-compose up command to build the container. We use docker desktop to see the status of the container. The container is shown in Figure 3.

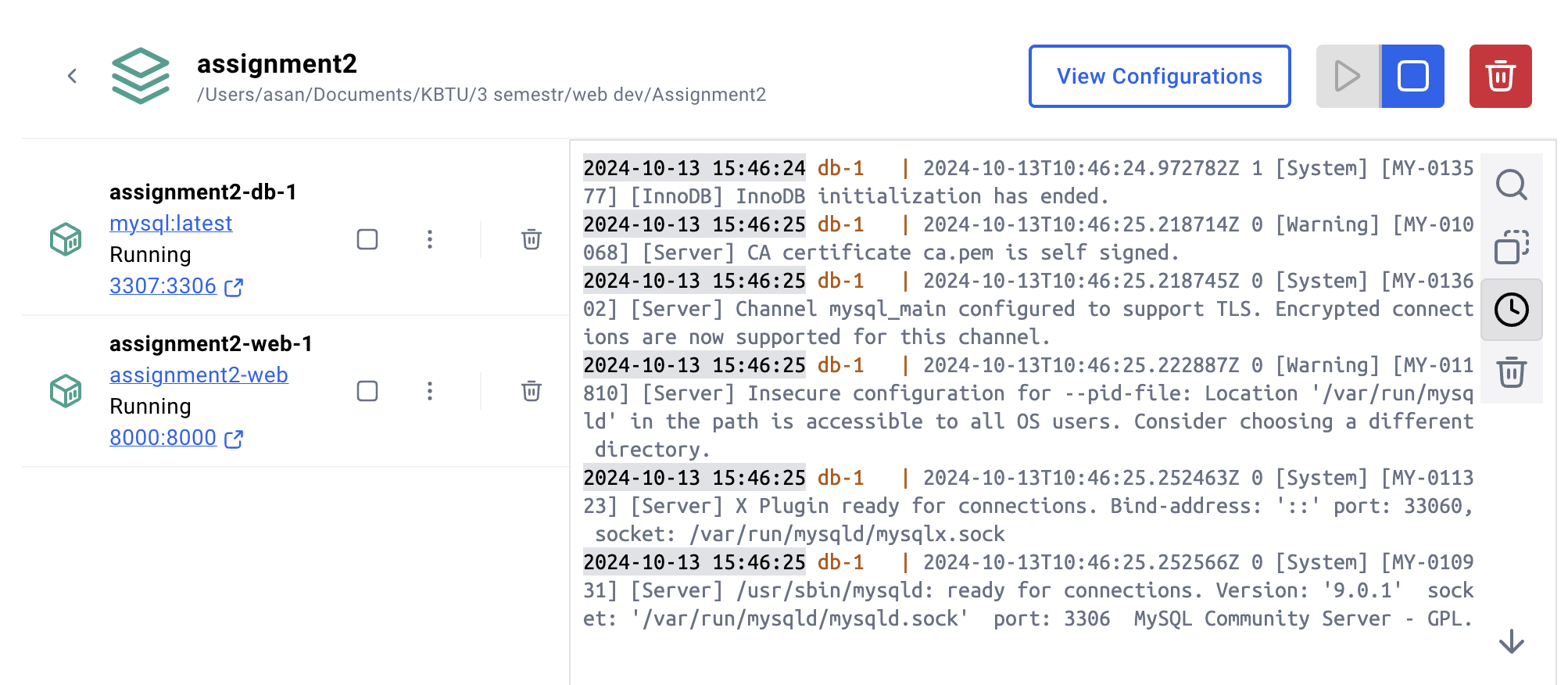


Figure 3. The assignment2 container status in docker desktop

We see that both db and web containers are running. The running web page is shown in Figure 4.

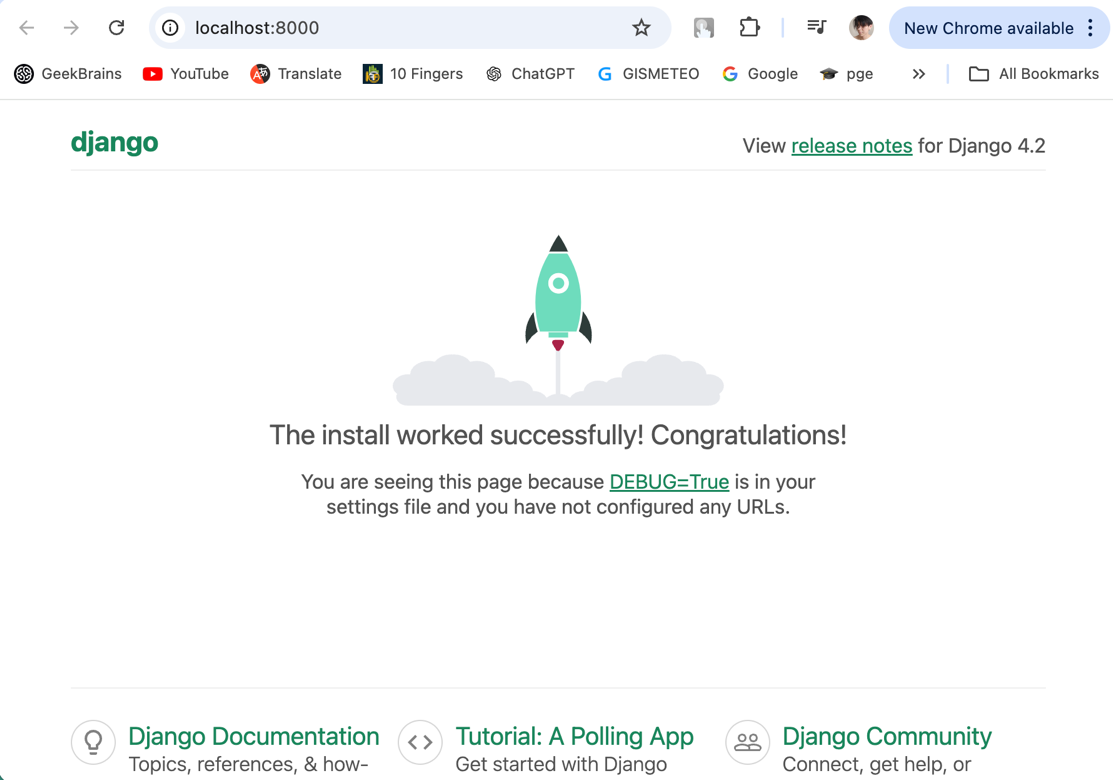


Figure 3. The assignment2 container status in docker desktop

This is a default page that was created by <django-admin startproject myproject .> command.

# Docker Networking and Volumes

Docker Networking allows containers to communicate with each other and with external systems by creating isolated networks. Docker Volumes allows us to save data from Docker containers in a way that keeps it safe outside the container, so the data doesn't get lost when the container is removed or updated. We define both of them in Figure 4.

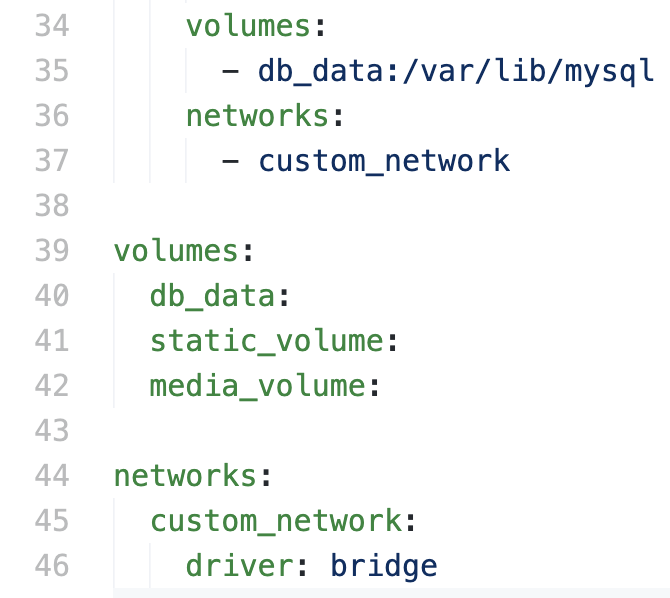


Figure 4. Defining the network and volumes

In this part of the docker-compose file we set up volumes and networks for Docker containers. The volumes section creates named storage areas like db\_data to save database files outside the container. The networks section, with custom\_network, connects the containers so they can communicate with each other. It uses a "bridge" driver, which creates an internal network for the containers to talk to each other securely.

Networking and volumes help the app run better. Networking lets the app and database talk to each other easily. Volumes save important data outside the container, so if the container stops, your data is still safe and can be used later.

# Django Application Setup

To create the blog page we create the blog directory in our project folder. Inside the foler we create two files: models.py and views.py. The code of these files shown in Figure 5.

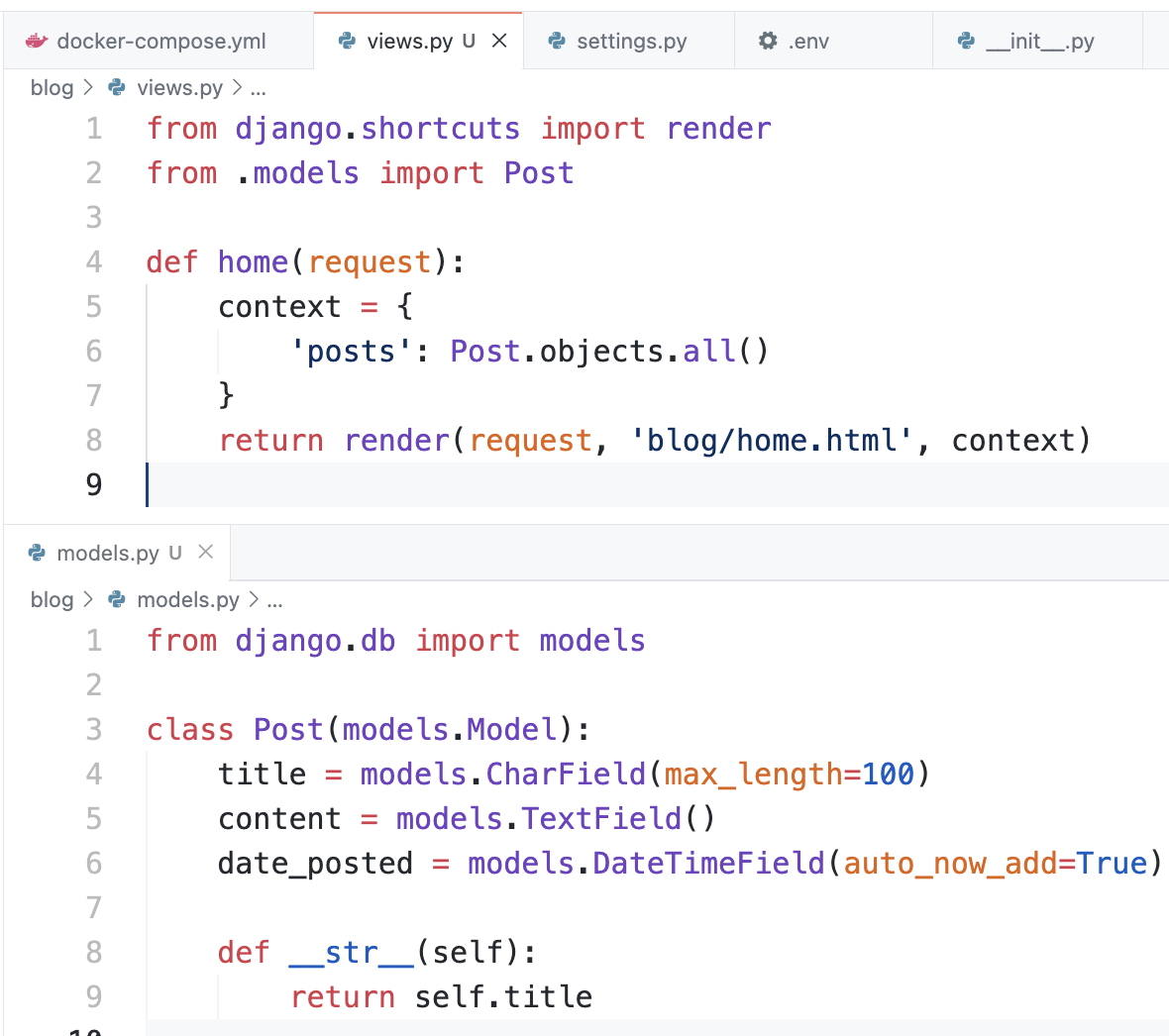


Figure 6. The code of models.py and views.py files

In the models file, we create a Post class that represents a blog post, with fields for the title, content, and the date it was posted. The \_\_str\_\_ method returns the title of the post when it is displayed, making it easier to identify in the admin interface.

In the views file, the home function retrieves all the posts from the database and stores them in a context dictionary. This context is then used to render the home.html template, displaying the list of blog posts to the user.

Then we run the Django app. The result of this is shown in Figure 6.

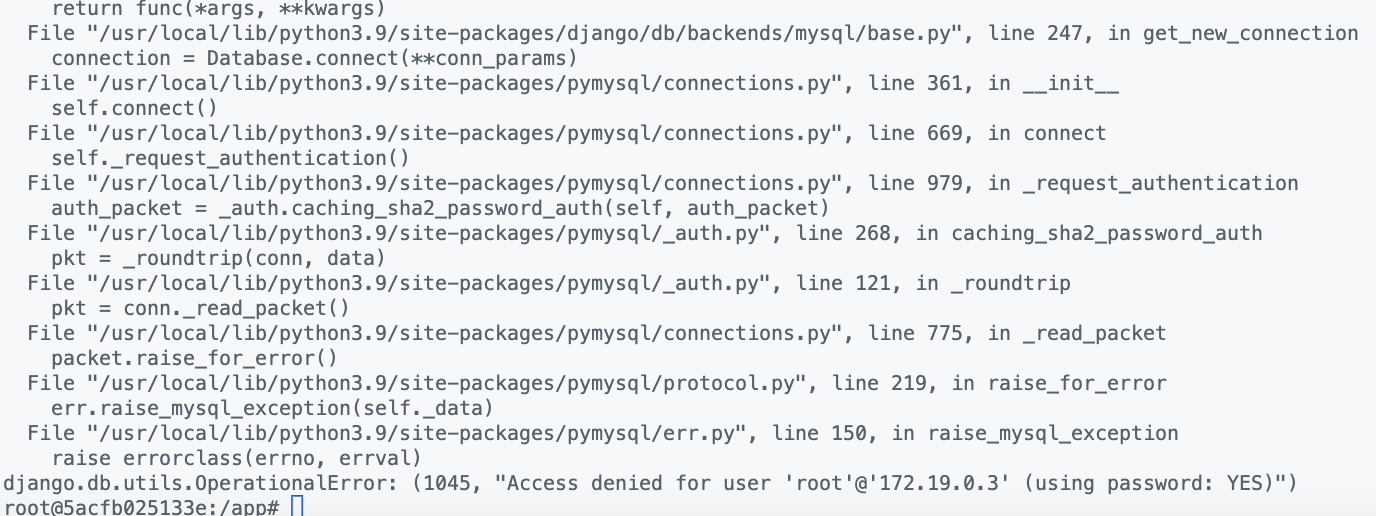


Figure 6. The output after running the project

There’s an error “django.db.utils.OperationalError: (1045, "Access denied for user 'root'@'172.19.0.3' (using password: YES)")”. We tried to fix the error by using configuring the sockets and host [1]. Then we tried to use IPv4Address for mysql container [2]. Also we tried to copy the file /conf/dist/.env.docker to .env and place it in the root folder of the project, then edit it. In the .env file we tried to configure: he location of the data, etc and logs folders, the open ports, the MySQL root password. None of these solutions didn’t help [3].

# Conclusion

In this assignment we learned about the theory of docker compose, networking and creation of django app. We learned the syntax and structure of docker-compose file, about how to create a container with both database and web app. Also we worked with volumes and ports. But in the end, we faced a major error that we couldn’t solve by the given deadline. We are going to keep finding the solution to be able to complete future assignments and tasks.

# References

1. Как на MacOS правильно подключить Django к БД в Docker? [Хабр] <https://qna.habr.com/q/1214856>
2. How can i connect django container to mysql container [StackOverflow] <https://stackoverflow.com/questions/69302349/how-can-i-connect-django-container-to-mysql-container>
3. How to change default MySQL username and password using docker setup? [StackOverflow] <https://stackoverflow.com/questions/68216560/how-to-change-default-mysql-username-and-password-using-docker-setup>