Assignment 2

Docker Compose

Name: Ibragimov Temirlan

Date of submission: October 13th

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

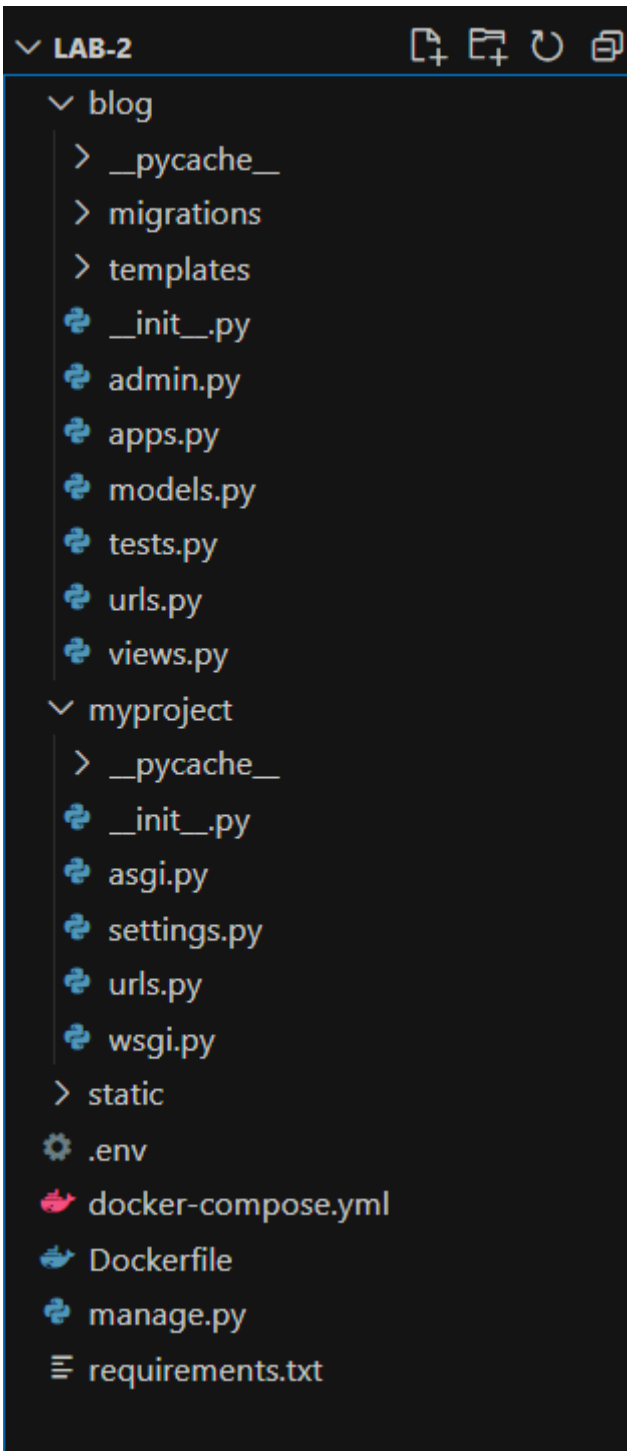
Configuration

```yaml
version: '3.9'

services:
  db:
    image: postgres
    environment:
      POSTGRES_DB: ${DB_NAME}
      POSTGRES_USER: ${DB_USER}
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data
    networks:
      - django_network

  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/code
      - static_volume:/code/static
    ports:
      - "8000:8000"
    environment:
      DB_NAME: ${DB_NAME}
      DB_USER: ${DB_USER}
      DB_PASSWORD: ${DB_PASSWORD}
      DB_HOST: db
      DB_PORT: 5432
    depends_on:
      - db
    networks:
      - django_network

volumes:
  postgres_data:
  static_volume:

networks:
  django_network:
```

Given docker-compose.yml file creates a multi-container Docker application with two services: django and postgres. It sets up a custom network named django_network as well.

Services

1. Database PostgreSQL
    a. Django uses the official postgres image.
    b. Configured with environment variables for database name, user, and password.
    c. Stores data using a volume (postgres_data).
    d. Connected to the django_network with the web service.
2. Django
    a. Builds the Django app from the local directory.
    b. Runs the server on port 8000.
    c. Uses volumes for code and static files.
    d. Depends on the db service for the database.
    e. Environment variables configure the connection to PostgreSQL.
3. Docker Settings
    a. postgres_data for persisting PostgreSQL data.
    b. static_volume for storing Django static files
    c. django_network connects the Django app and the PostgreSQL database.



Build and run

After writing Dockerfiles and other code, you need to run the "docker compose –build" command, and then the "docker compose up" command.

Docker Networking and Volumes

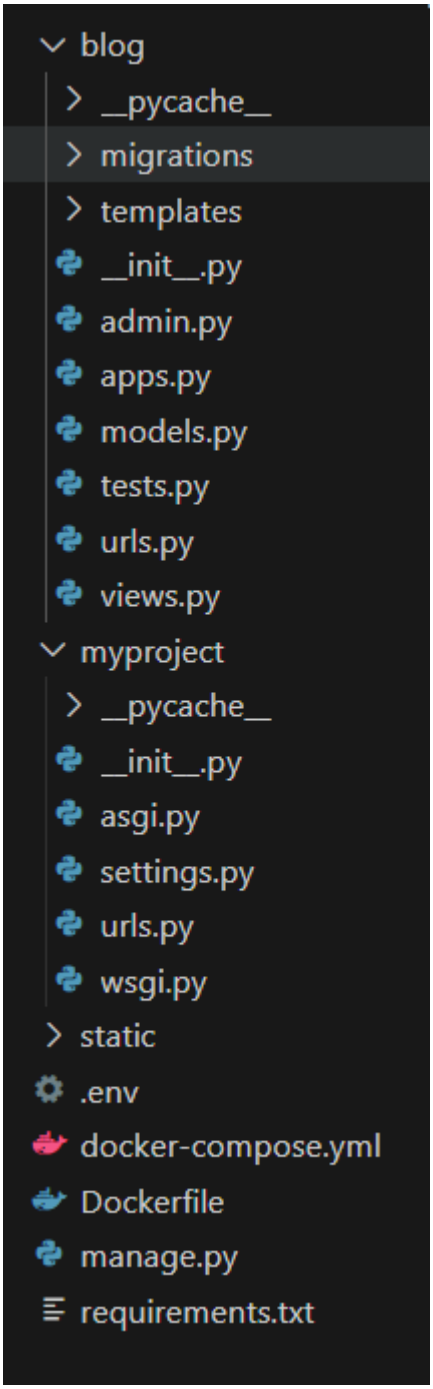
- **Custom Network Setup and Its Benefits**
  The docker-compose.yml file defines a custom network named net using the bridge driver. This network facilitates communication between the django and postgres services. Here are the key benefits of this setup:
    - **Isolation**: The custom network isolates the services from other containers running on the host, enhancing security.
    - **Service Discovery**: Docker automatically assigns each service a hostname based on the service name, making it easy for services to discover and communicate with each other.
    - **Simplified Configuration**: By using a custom network, you avoid the need to manually configure IP addresses and ports for inter-service communication.
- **Volumes for Data Persistence**
  Volumes are used in the docker-compose.yml file to ensure data persistence for both the django and postgres services:
    - **Django Service**:
        - The volume ./django:/usr/src/app mounts the ./django directory on the host to /usr/src/app in the container. This allows the Django application code to be accessible and editable from the host machine.
    - **Postgres Service**:
        - The volume ./postgres/data:/var/lib/postgresql/data mounts the ./postgres/data directory on the host to /var/lib/postgresql/data in the container. This ensures that the PostgreSQL database data is stored persistently on the host, even if the container is stopped or removed.
- **Advantages of Using Docker Networking and Volumes**
  Using Docker networking and volumes in your application offers several advantages:
    - **Networking**:
        - **Enhanced Security**: Custom networks isolate containers, reducing the risk of unauthorized access.

        - **Ease of Communication**: Services can easily communicate using service names, simplifying configuration.
        - **Scalability**: Networks can be easily scaled to include more services or containers as needed.
    - **Volumes**:
        - **Data Persistence**: Volumes ensure that data is not lost when containers are stopped or removed.
        - **Separation of Concerns**: Application code and data are managed separately, making it easier to update and maintain the application.
        - **Performance**: Volumes are optimized for performance, providing faster data access compared to bind mounts.

These features make Docker a powerful tool for developing, deploying, and managing applications in a consistent and efficient manner.

Django application setup

Creating Django Project



Post Model

```python
from django.db import models
from django.utils import timezone

class Post(models.Model):
    title = models.CharField(max_length=200)
    content = models.TextField()
    author = models.CharField(max_length=100)
    created_at = models.DateTimeField(default=timezone.now)

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

Configure the Database

myproject > settings.py > ...

```
                'context_processors': [
                    'django.template.context_processors.debug',
                    'django.template.context_processors.request',
                    'django.contrib.auth.context_processors.auth',
                    'django.contrib.messages.context_processors.messages',
                ],
            },
        },
    ]

    WSGI_APPLICATION = 'myproject.wsgi.application'


    # Database
    # https://docs.djangoproject.com/en/5.1/ref/settings/#databases

    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.postgresql',
            'NAME': os.getenv('DB_NAME'),
            'USER': os.getenv('DB_USER'),
            'PASSWORD': os.getenv('DB_PASSWORD'),
            'HOST': os.getenv('DB_HOST', 'db'),
            'PORT': '5432',
        }
    }

```

```
PS D:\University\WebApp\Assignments\lab-2> docker compose up
time="2024-10-13T23:25:56+05:00" level=warning msg="D:\\University\\WebA
nored, please remove it to avoid potential confusion"
[+] Running 2/0
 ✓ Container lab-2-db-1   Created
 ✓ Container lab-2-web-1  Created
Attaching to db-1, web-1
db-1   |
db-1   | PostgreSQL Database directory appears to contain a database; Sk
db-1   |
db-1   | 2024-10-13 18:25:56.476 UTC [1] LOG:  starting PostgreSQL 17.0
4-bit
db-1   | 2024-10-13 18:25:56.476 UTC [1] LOG:  listening on IPv4 address
db-1   | 2024-10-13 18:25:56.476 UTC [1] LOG:  listening on IPv6 address
db-1   | 2024-10-13 18:25:56.499 UTC [1] LOG:  listening on Unix socket
db-1   | 2024-10-13 18:25:56.530 UTC [29] LOG:  database system was shut
db-1   | 2024-10-13 18:25:56.542 UTC [1] LOG:  database system is ready
web-1  | Watching for file changes with StatReloader
db-1   | 2024-10-13 18:26:00.359 UTC [34] ERROR:  relation "django_sessi
db-1   | 2024-10-13 18:26:00.359 UTC [34] STATEMENT:  SELECT "django_ses
ngo_session" WHERE ("django_session"."expire_date" > '2024-10-13T18:26:0
9zdpd') LIMIT 21
db-1   | 2024-10-13 18:26:00.392 UTC [34] ERROR:  relation "django_sessi
db-1   | 2024-10-13 18:26:00.392 UTC [34] STATEMENT:  SELECT "django_ses
ngo_session" WHERE ("django_session"."expire_date" > '2024-10-13T18:26:0
9zdpd') LIMIT 21
db-1   | 2024-10-13 18:26:00.394 UTC [34] ERROR:  relation "django_sessi
db-1   | 2024-10-13 18:26:00.394 UTC [34] STATEMENT:  SELECT "django_ses
ngo_session" WHERE ("django_session"."expire_date" > '2024-10-13T18:26:0
9zdpd') LIMIT 21
db-1   | 2024-10-13 18:26:00.400 UTC [34] ERROR:  relation "django_sessi
db-1   | 2024-10-13 18:26:00.400 UTC [34] STATEMENT:  SELECT "django_ses
ngo_session" LIMIT 21
```

Conclusion

Integration of Docker and Django

This assignment highlighted the effective integration of Docker with Django, focusing on its benefits for environment management and deployment consistency. Key takeaways include the importance of containerization for maintaining isolated environments, simplifying dependencies, and ensuring portability across various systems. Docker enhances development workflows, streamlines collaboration, and improves scalability for Django applications through the use of Docker

Compose, which simplifies the setup and deployment of multi-container applications.

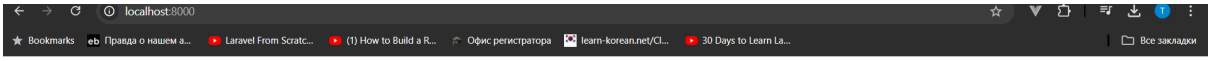
Importance of Health Checks

Implementing health checks for containers, especially those running databases, is crucial for application reliability. Health checks ensure that the database service is fully operational before dependent services, like Django, start, preventing potential errors and ensuring a smooth startup process. Overall, using Docker with Django simplifies development and deployment by encapsulating the application and its dependencies in containers, allowing for consistent operation across different environments and easier scaling of application components.

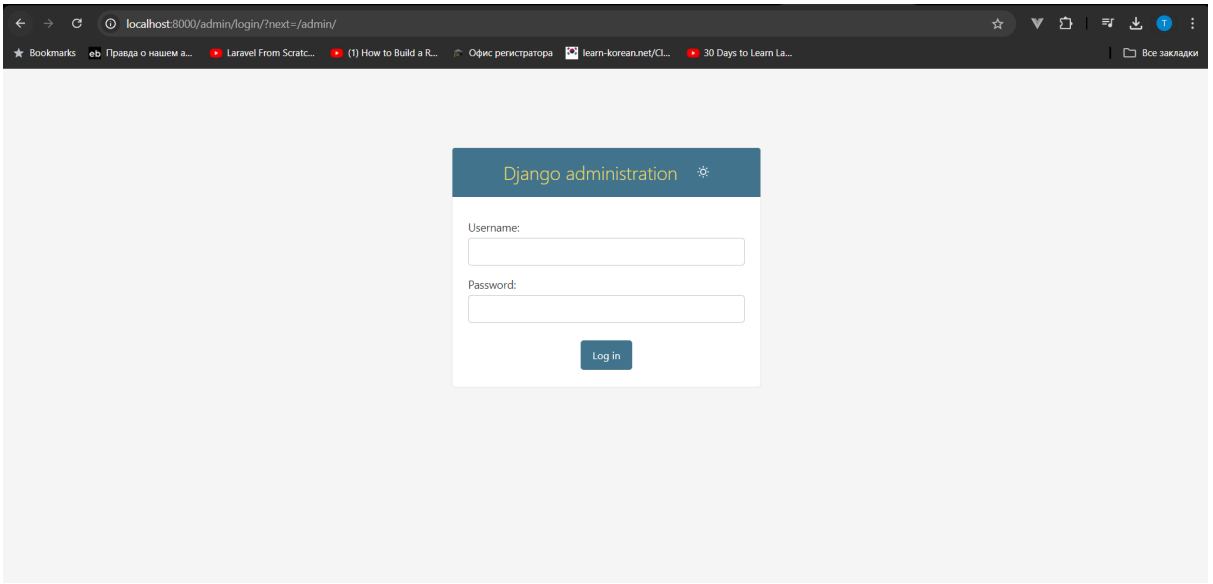
References

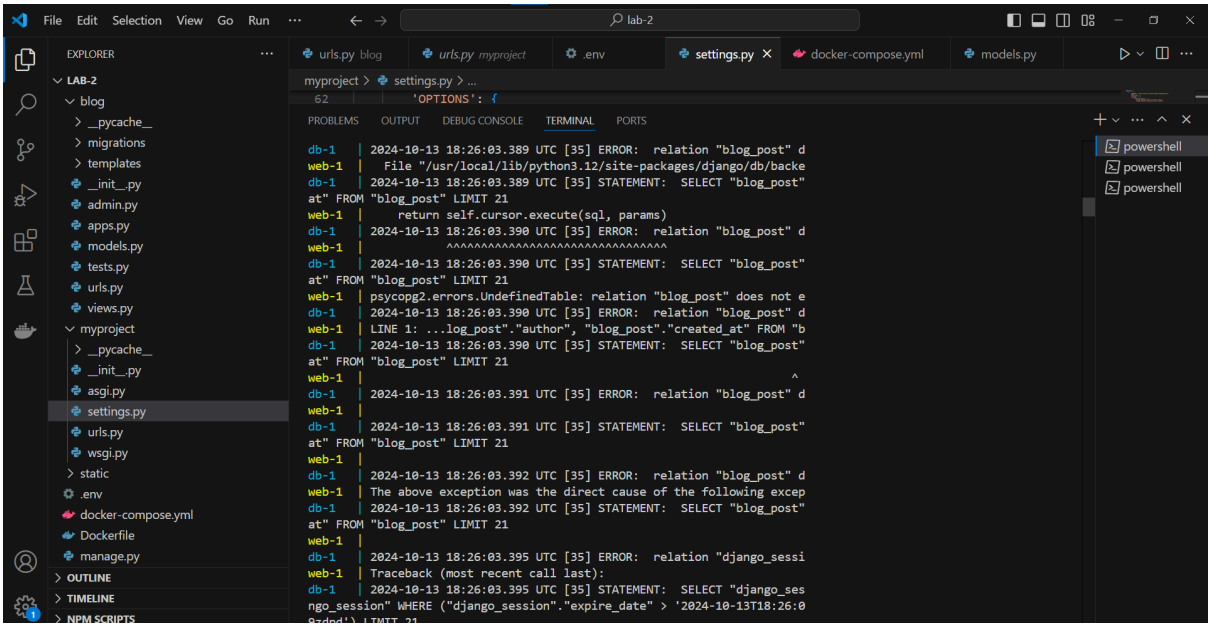
https://docs.docker.com/
https://docs.djangoproject.com/en/5.1/

Screenshots



Blog Posts






```
PS D:\University\WebApp\Assignments\lab-2> docker-compose exec web python manage.py migrate
time="2024-10-13T23:28:20+05:00" level=warning msg="D:\\University\\WebApp\\Assignments\\lab-2\\docker-compose.y
te, it will be ignored, please remove it to avoid potential confusion"
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.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 blog.0001_initial... OK
  Applying sessions.0001_initial... OK
PS D:\University\WebApp\Assignments\lab-2> docker-compose exec web python manage.py makemigrations
te, it will be ignored, please remove it to avoid potential confusion"
No changes detected
PS D:\University\WebApp\Assignments\lab-2> docker-compose exec web python manage.py migrate
time="2024-10-13T23:30:11+05:00" level=warning msg="D:\\University\\WebApp\\Assignments\\lab-2\\docker-compose.y
te, it will be ignored, please remove it to avoid potential confusion"
Operations to perform:
```