KAZAKH-BRITISH TECHNICAL UNIVERSITY

KBTU

Assignment 2, Web app dev
Title: Exploring Django with Docker
Student: Ramazan Aliyev
Date of submission: 13.10.2024

Table of Contents

Introduction

Instruction:

Put all deliverables into github repository in your profile. Share link to google form according to teams deadline. Defend by explaining deliverables and answering questions.
Deliverables: report in pdf
Google form: https://docs.google.com/forms/d/e/1FAIpQLSe0GyNdOYlvM1tX_I_CtlPod5jBf-ACLGdHYZq1gVZbUeBzIg/viewform?usp=sf_link

Git: https://github.com/rvmzik/Lab2_Web.git

Objective

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.

Deliverables

1. A fully functional Django application running in a Docker container.
2. A detailed report covering:
   - Docker Compose
   - Docker Networking and Volumes
   - Django application setup
3. Screenshots or code snippets demonstrating configurations and setups.

There are 3 tasks.

There are:

Docker Compose

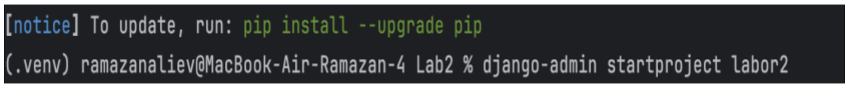
Docker Networking and Volumes

Django Application Setup

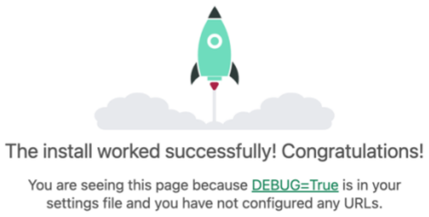
1. Docker Compose

- **Create a Docker Compose File**
  - Create a `docker-compose.yml` file for your Django application.

    Firstly I created a django project and checked it



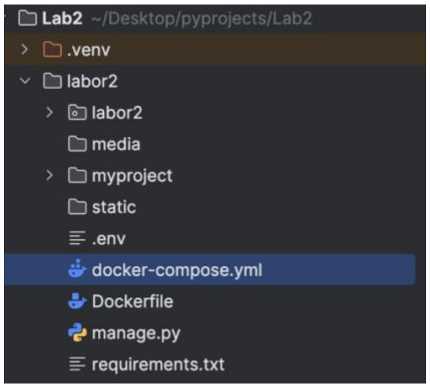
```
[notice] To update, run: pip install --upgrade pip
(.venv) ramazanaliev@MacBook-Air-Ramazan-4 Lab2 % django-admin startproject labor2
```



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.

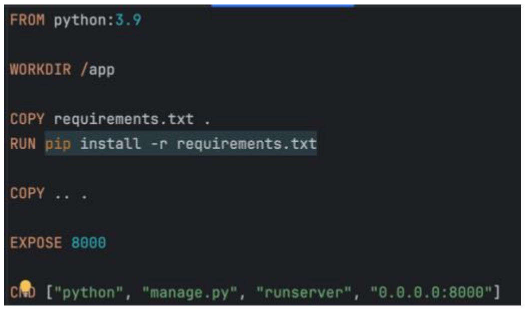
Then I created a docker-compose.yml file inside project the project. You can see it below



```
Lab2 ~/Desktop/pyprojects/Lab2
  > .venv
  v labor2
      > labor2
        media
      > myproject
        static
        .env
        docker-compose.yml
        Dockerfile
        manage.py
        requirements.txt
```

- Include services for:
  - Django web server
  - PostgreSQL database (or another database of your choice)

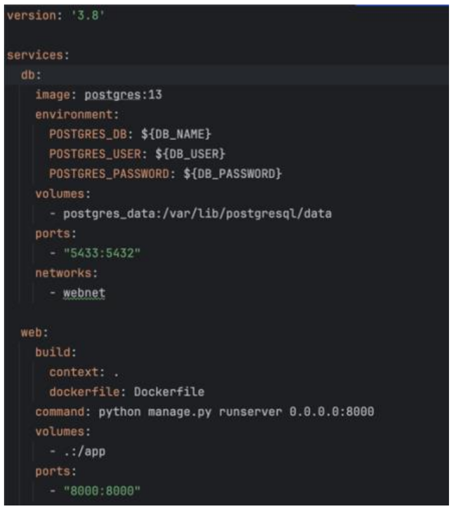
  I created Dockerfile to create an image, then below

```
FROM python:3.9

WORKDIR /app

COPY requirements.txt .
RUN pip install -r requirements.txt

COPY .. .

EXPOSE 8000

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

- I chose PostgreSQL database and write environment and chose port 5433 because there was 5432 in use and chose 13 postgres. I defined 2 services.

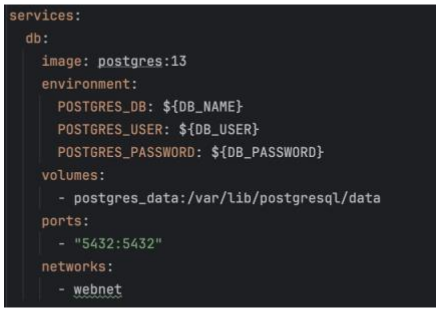
  There are db and web. Where ports 5433 and 8000

- 

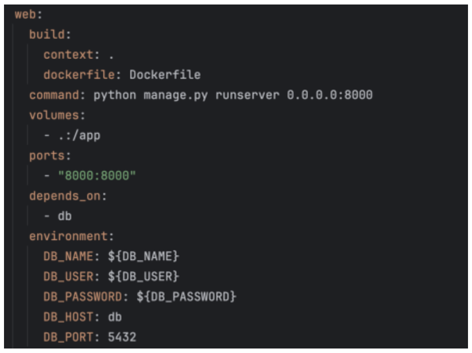
```
version: '3.8'

services:
  db:
    image: postgres:13
    environment:
      POSTGRES_DB: ${DB_NAME}
      POSTGRES_USER: ${DB_USER}
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5433:5432"
    networks:
      - webnet

  web:
    build:
      context: .
      dockerfile: Dockerfile
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/app
    ports:
      - "8000:8000"
```

- **Define Environment Variables**
  - Use environment variables for database configuration (e.g., DB_NAME, DB_USER, DB_PASSWORD).

  Like I said I added environment to 2 services:

  - 

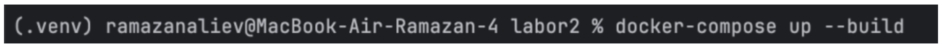
```
services:
  db:
    image: postgres:13
    environment:
      POSTGRES_DB: ${DB_NAME}
      POSTGRES_USER: ${DB_USER}
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5432:5432"
    networks:
      - webnet
```



```
web:
  build:
    context: .
    dockerfile: Dockerfile
  command: python manage.py runserver 0.0.0.0:8000
  volumes:
    - .:/app
  ports:
    - "8000:8000"
  depends_on:
    - db
  environment:
    DB_NAME: ${DB_NAME}
    DB_USER: ${DB_USER}
    DB_PASSWORD: ${DB_PASSWORD}
    DB_HOST: db
    DB_PORT: 5432
```

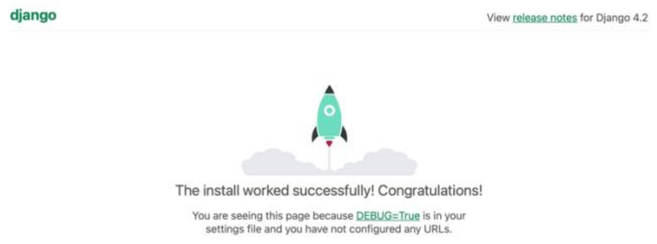
- **Build and Run the Containers**
  - Use `docker-compose up` to build and run the application.

    I write docker-compose up –build and it was successful and then I checked it



```
(.venv) ramazanaliev@MacBook-Air-Ramazan-4 labor2 % docker-compose up --build
```

  - Ensure that the services are running correctly

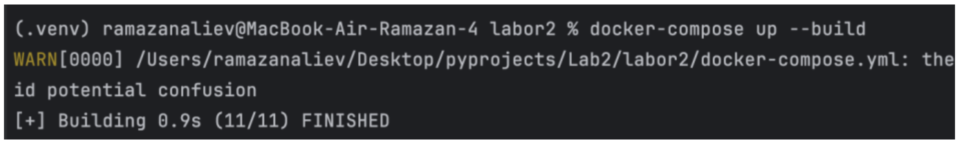


django

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.

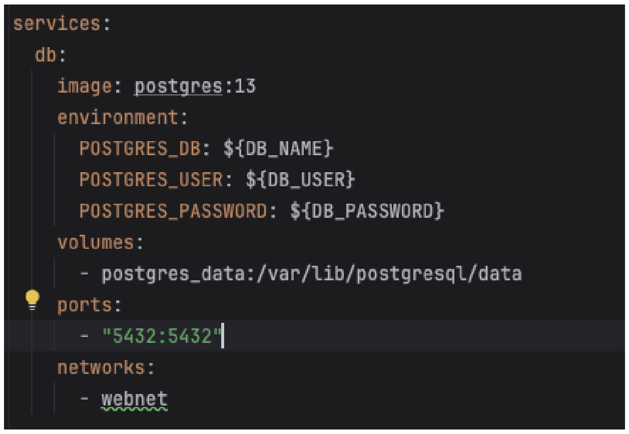


```
(.venv) ramazanaliev@MacBook-Air-Ramazan-4 labor2 % docker-compose up --build
WARN[0000] /Users/ramazanaliev/Desktop/pyprojects/Lab2/labor2/docker-compose.yml: the
id potential confusion
[+] Building 0.9s (11/11) FINISHED
```

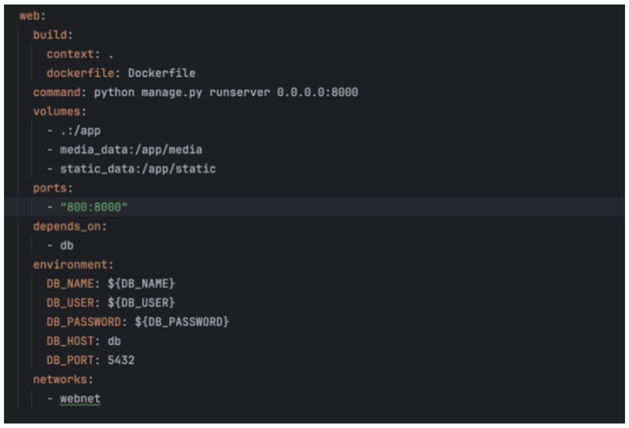
2. Docker Networking and Volumes

- **Set Up Docker Networking**
  - Define a custom network in your `docker-compose.yml` file to allow communication between services.
  -

  As you can see I added webnet network. 2 services are connected to that network.



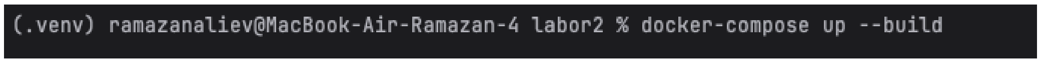
```
services:
  db:
    image: postgres:13
    environment:
      POSTGRES_DB: ${DB_NAME}
      POSTGRES_USER: ${DB_USER}
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5432:5432"
    networks:
      - webnet
```



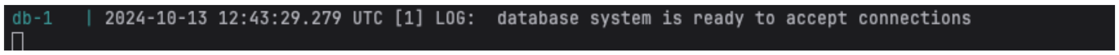
```
web:
  build:
    context: .
    dockerfile: Dockerfile
  command: python manage.py runserver 0.0.0.0:8000
  volumes:
    - .:/app
    - media_data:/app/media
    - static_data:/app/static
  ports:
    - "800:8000"
  depends_on:
    - db
  environment:
    DB_NAME: ${DB_NAME}
    DB_USER: ${DB_USER}
    DB_PASSWORD: ${DB_PASSWORD}
    DB_HOST: db
    DB_PORT: 5432
  networks:
    - webnet
```

  - Verify that the Django app can connect to the database using the network.

    Firstly, I build an app and checked it.

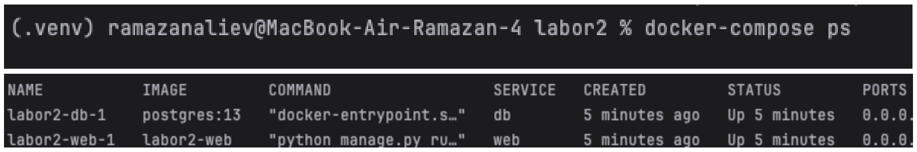


```
(.venv) ramazanaliev@MacBook-Air-Ramazan-4 labor2 % docker-compose up --build
```

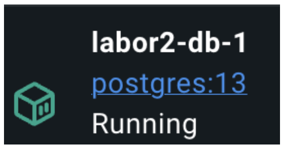


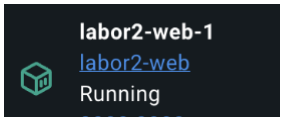
```
db-1    | 2024-10-13 12:43:29.279 UTC [1] LOG:  database system is ready to accept connections
```

    Then I write docker-compose ps to get info about containers and checked it


```
(.venv) ramazanaliev@MacBook-Air-Ramazan-4 labor2 % docker-compose ps
```
```
NAME            IMAGE          COMMAND                   SERVICE    CREATED           STATUS          PORTS
labor2-db-1     postgres:13    "docker-entrypoint.s…"    db         5 minutes ago     Up 5 minutes    0.0.0.
labor2-web-1    labor2-web     "python manage.py ru…"    web        5 minutes ago     Up 5 minutes    0.0.0.
```

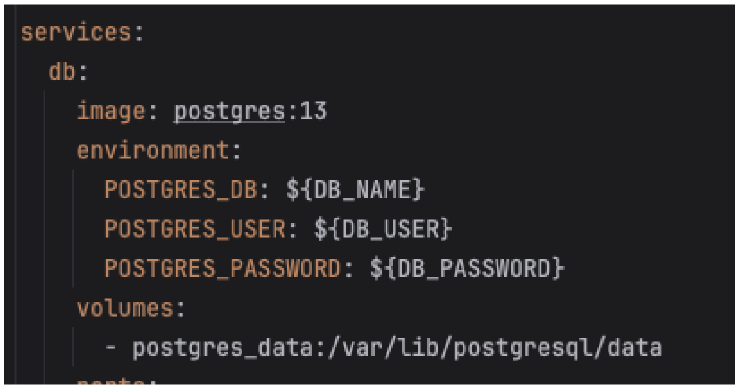
Then I check Docker desktop to get results


labor2-db-1
postgres:13
Running

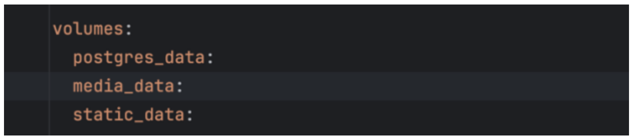

labor2-web-1
labor2-web
Running

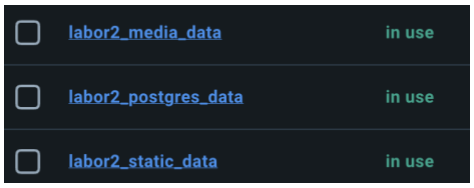
- **Implement Docker Volumes**
    - Configure a volume in the `docker-compose.yml` file to persist PostgreSQL data.

      I added volumes to services and defined them in volumes


```yaml
services:
  db:
    image: postgres:13
    environment:
      POSTGRES_DB: ${DB_NAME}
      POSTGRES_USER: ${DB_USER}
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data
```

I added media and static data to save uploaded files by users and static files.

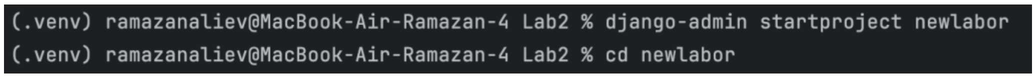

```yaml
volumes:
  postgres_data:
  media_data:
  static_data:
```


☐	labor2_media_data	in use
☐	labor2_postgres_data	in use
☐	labor2_static_data	in use

3. Django Application Setup

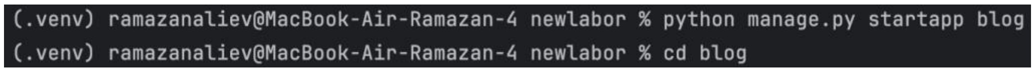
- **Create a Django Project**
  - Inside the Django service container, create a new Django project using the command `django-admin startproject myproject`.

    Firstly I created a new project with name newlabor



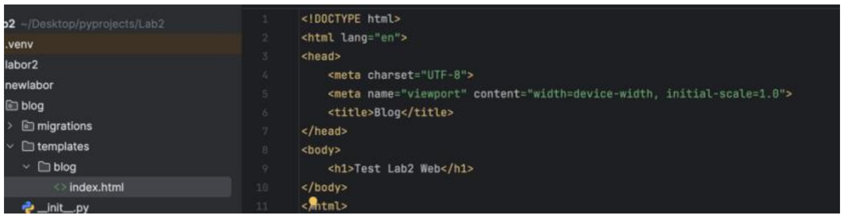
```
(.venv) ramazanaliev@MacBook-Air-Ramazan-4 Lab2 % django-admin startproject newlabor
(.venv) ramazanaliev@MacBook-Air-Ramazan-4 Lab2 % cd newlabor
```

  - Create a simple app (e.g., `blog`) with at least one model and a corresponding view.
    Then I created a simple app in the newlabor which name is blog



```
(.venv) ramazanaliev@MacBook-Air-Ramazan-4 newlabor % python manage.py startapp blog
(.venv) ramazanaliev@MacBook-Air-Ramazan-4 newlabor % cd blog
```

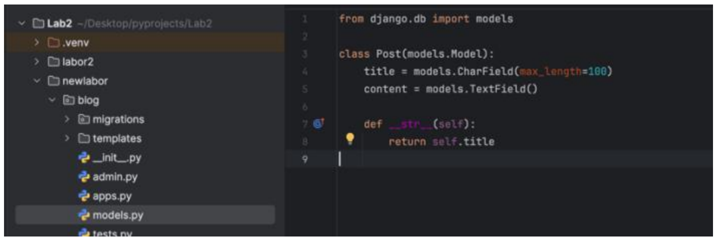
    In blog I defined templates and added index in blog.
    In index I defined Test Lab2 Web



```
2 ~/Desktop/pyprojects/Lab2
.venv
labor2
newlabor
  blog
    migrations
    templates
      blog
        index.html
    __init__.py
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog</title>
</head>
<body>
    <h1>Test Lab2 Web</h1>
</body>
</html>
```

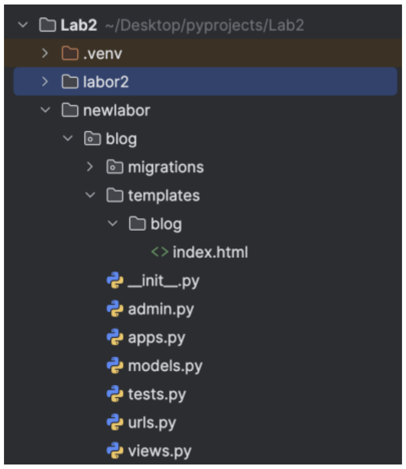
    And create one model



```
Lab2 ~/Desktop/pyprojects/Lab2
  .venv
  labor2
  newlabor
    blog
      migrations
      templates
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
```

```python
from django.db import models

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

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

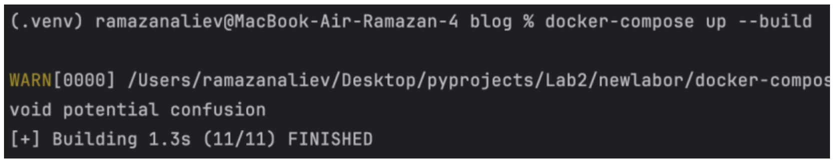
    Then I got this

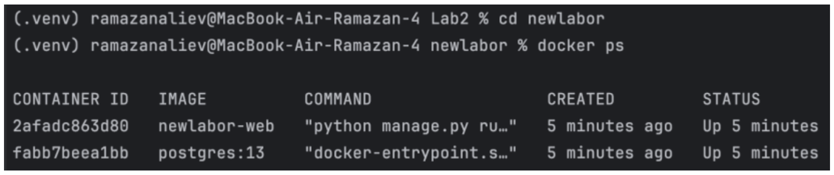


```
Lab2 ~/Desktop/pyprojects/Lab2
  .venv
  labor2
  newlabor
    blog
      migrations
      templates
        blog
          index.html
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      urls.py
      views.py
```

- **Configure the Database**
  - Update the Django settings to use the PostgreSQL database configured in your Docker Compose setup.

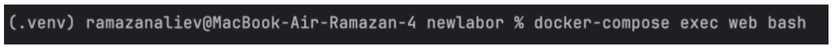
I build an app and check running containers


```
(.venv) ramazanaliev@MacBook-Air-Ramazan-4 blog % docker-compose up --build

WARN[0000] /Users/ramazanaliev/Desktop/pyprojects/Lab2/newlabor/docker-compos
void potential confusion
[+] Building 1.3s (11/11) FINISHED
```

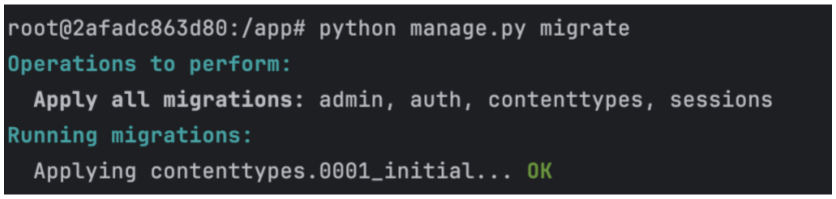

```
(.venv) ramazanaliev@MacBook-Air-Ramazan-4 Lab2 % cd newlabor
(.venv) ramazanaliev@MacBook-Air-Ramazan-4 newlabor % docker ps

CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS
2afadc863d80   newlabor-web   "python manage.py ru…"   5 minutes ago   Up 5 minutes
fabb7beea1bb   postgres:13    "docker-entrypoint.s…"   5 minutes ago   Up 5 minutes
```

- Run migrations to set up the database schema.

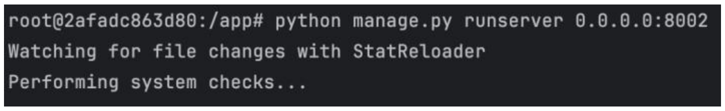
Firstly I added exec web base to be able migrate and runserver

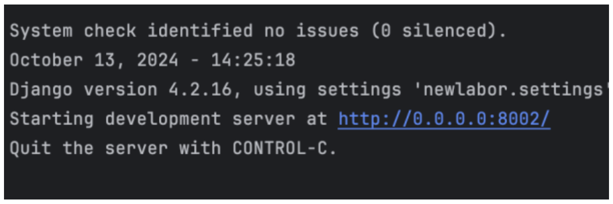

```
(.venv) ramazanaliev@MacBook-Air-Ramazan-4 newlabor % docker-compose exec web bash
```

Then I applied migration

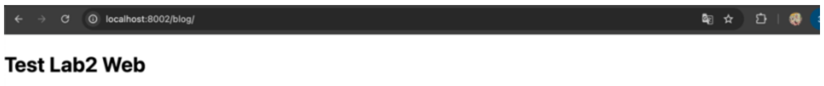

```
root@2afadc863d80:/app# python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
```

Then I run server


```
root@2afadc863d80:/app# python manage.py runserver 0.0.0.0:8002
Watching for file changes with StatReloader
Performing system checks...
```


```
System check identified no issues (0 silenced).
October 13, 2024 - 14:25:18
Django version 4.2.16, using settings 'newlabor.settings'
Starting development server at http://0.0.0.0:8002/
Quit the server with CONTROL-C.
```

Final I got result:


localhost:8002/blog/

Test Lab2 Web

Conclusion

In this assignment I learned how to work with docker-compose.yml. How to work with Django and testing and checking results and work with migrations.
Significance:
I think the significance in in convenient application deployment and development. It can simplify the process. In addition to scale applications.

References

https://hub.docker.com/
https://habr.com/ru/companies/ruvds/articles/450312/
https://docs.docker.com/compose/