

Департамент образования и науки города Москвы  
Государственное автономное образовательное учреждение  
высшего образования города Москвы  
«Московский городской педагогический университет»  
Институт цифрового образования  
Департамент информатики управления и технологий

Нургалеева Гузель Рустэмовна БД-241м

**Практическая работа 3-2. Docker compose**

Направление подготовки/специальность  
38.04.05 - Бизнес-информатика  
Бизнес-аналитика и большие данные  
(очная форма обучения)

Москва

2024

## Задача

Создать docker-compose.yml из минимум трех сервисов

**На основе Dockerfile из Практическая работа 3.1. создать композ проект. Обязательные требования:**

- минимум 1 init + 2 app сервиса (одноразовый init + приложение + бд или что-то другое, главное чтоб работало в связке)
- автоматическая сборка образа из лежащего рядом Dockerfile и присваивание ему (образу) имени
- жесткое именование получившихся контейнеров
- минимум один из сервисов обязательно с depends\_on
- минимум один из сервисов обязательно с volume
- минимум один из сервисов обязательно с прокидыванием порта наружу
- минимум один из сервисов обязательно с ключом command и/или Entrypoint (можно переиспользовать тот же, что в Dockerfile)
- добавить healthcheck
- все env-ы прописать не в сам docker-compose.yml, а в лежащий рядом файл .env
- должна быть явно указана network (одна для всех)

## Запуск

```
cd lab3_2
cp .env.example .env
docker-compose up
OpenAPI:
```

`http://localhost:8000/api/docs`

Health Check:

<http://localhost:8000/health>

## 1. Описание проекта.

- Кратко опишите цель работы и общий смысл архитектуры (init, app, db сервисы и их взаимодействие).

**Цель:** понять структуру проекта, назначение файлов в нем (за что отвечают, какие есть важные настройки).

### Архитектура:

dba@dba-vm:~/DevOps-Technologies/Lesson 3 Containerizing Applications/lab3\_2\$ tree

```
.
├── client
│   ├── Dockerfile
│   ├── index.html
│   ├── package.json
│   ├── package-lock.json
│   ├── postcss.config.cjs
│   ├── public
│   │   └── vite.svg
│   ├── README.md
│   └── src
│       ├── assets
│       │   └── react.svg
│       ├── components
│       │   ├── About
│       │   │   ├── About.module.css
│       │   │   ├── About.tsx
│       │   │   └── index.ts
│       │   ├── Achievements
│       │   │   ├── Achievements.module.css
│       │   │   ├── Achievements.tsx
│       │   │   └── index.ts
│       │   ├── Actions
│       │   │   ├── Actions.module.css
│       │   │   ├── Actions.tsx
│       │   │   └── index.ts
│       │   ├── AuthForm
│       │   │   ├── AuthForm.module.css
│       │   │   ├── AuthForm.tsx
│       │   │   └── index.ts
│       │   ├── common
│       │   │   ├── Header
│       │   │   │   ├── Header.module.css
│       │   │   │   ├── Header.tsx
│       │   │   │   └── index.ts
│       │   │   ├── index.ts
│       │   │   ├── Logo.tsx
│       │   │   ├── NothingFound
│       │   │   │   └── Illustration.tsx
│       │   │   └── index.ts
```

```

graph TD
    src[ ] --- components[ ]
    src --- hooks[ ]
    src --- skills[ ]
    src --- teammate[ ]
    src --- teammates[ ]
    src --- teammateThumbnail[ ]
    src --- lib[ ]
    src --- main[ ]
    src --- viteEnvD[vite-env.d.ts]
    src --- tsconfig[ ]
    src --- viteConfig[vite.config.ts]
    src --- dockerCompose[docker-compose.yml]
    src --- docs[ ]
    src --- architecture[architecture.png]
    src --- logo[Hahathon Logo.png]

    components --- NothingFound[NothingFound.module.css  
NothingFound.tsx]
    components --- ThemeSwitch[ThemeSwitch  
index.ts  
ThemeSwitch.module.css  
ThemeSwitch.tsx]
    components --- UserMenu[UserMenu  
index.ts  
UserMenu.module.css  
UserMenu.tsx]
    hooks --- hoc[hoc  
index.ts  
withHeader.tsx]
    hooks --- useToken[useToken.tsx]
    skills --- skills[Skills  
index.ts  
Skills.module.css  
Skills.tsx]
    teammate --- teammate[Teammate  
index.ts  
Teammate.module.css  
Teammate.tsx]
    teammates --- teammates[Teammates  
index.ts  
Teammates.module.css  
Teammates.tsx]
    teammateThumbnail --- teammateThumbnail[TeammateThumbnail  
index.ts  
TeammateThumbnail.module.css  
TeammateThumbnail.tsx]
    lib --- lib[lib  
api.ts  
state-engine.ts]
    main --- main[main.module.css  
main.tsx  
routers.tsx  
routes  
auth.tsx  
root.tsx]
    viteEnvD --- viteEnvD[vite-env.d.ts]
    tsconfig --- tsconfig[tsconfig.json  
tsconfig.node.json]
    viteConfig --- viteConfig[vite.config.ts]
    dockerCompose --- dockerCompose[docker-compose.yml]
    docs --- docs[docs]
    architecture --- architecture[architecture.png]
    logo --- logo[Hahathon Logo.png]
  
```

```
|
| |
| | ├── main_page.png
| | ├── person_card.png
| | ├── profile_modal_page.png
| | ├── start_page.png
| | └── teams_page.png
| |
| ├── __init__.py
| ├── nginx
| | ├── Dockerfile
| | └── nginx.conf.template
| ├── README.md
| ├── requirements.txt
| └── server
|     ├── alembic.ini
|     ├── Dockerfile
|     ├── __init__.py
|     ├── requirements.txt
|     └── src
|         ├── app.py
|         ├── auth
|         | ├── auth.py
|         | ├── enums.py
|         | ├── __init__.py
|         | ├── routers.py
|         | ├── schemas.py
|         | ├── services.py
|         | └── utils.py
|         ├── common
|         | ├── __init__.py
|         | └── routers.py
|         ├── core
|         | ├── database.py
|         | ├── __init__.py
|         | ├── models.py
|         | ├── routers.py
|         | ├── schemas.py
|         | └── settings.py
|         ├── __init__.py
|         ├── main.py
|         ├── migrations
|         | ├── env.py
|         | ├── __init__.py
|         | ├── script.py.mako
|         | └── versions
|         |     ├── ea82a03adddfc_initial.py
|         |     └── __init__.py
|         └── users
|             ├── enums.py
|             ├── __init__.py
|             └── models.py
```

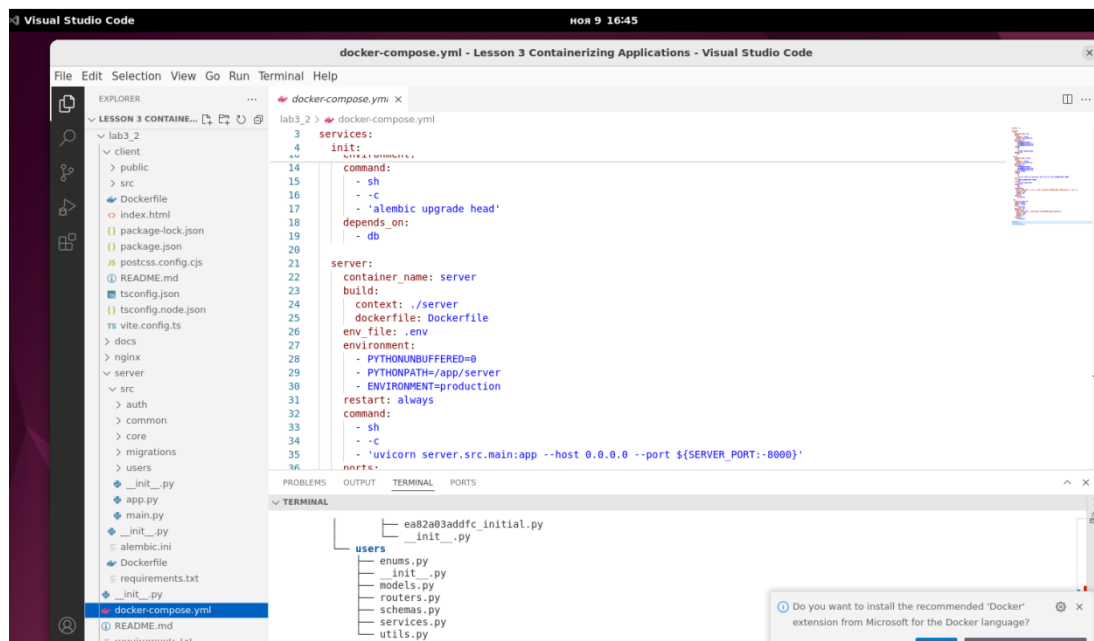
```
├── routers.py
├── schemas.py
├── services.py
└── utils.py
```

32 directories, 109 files

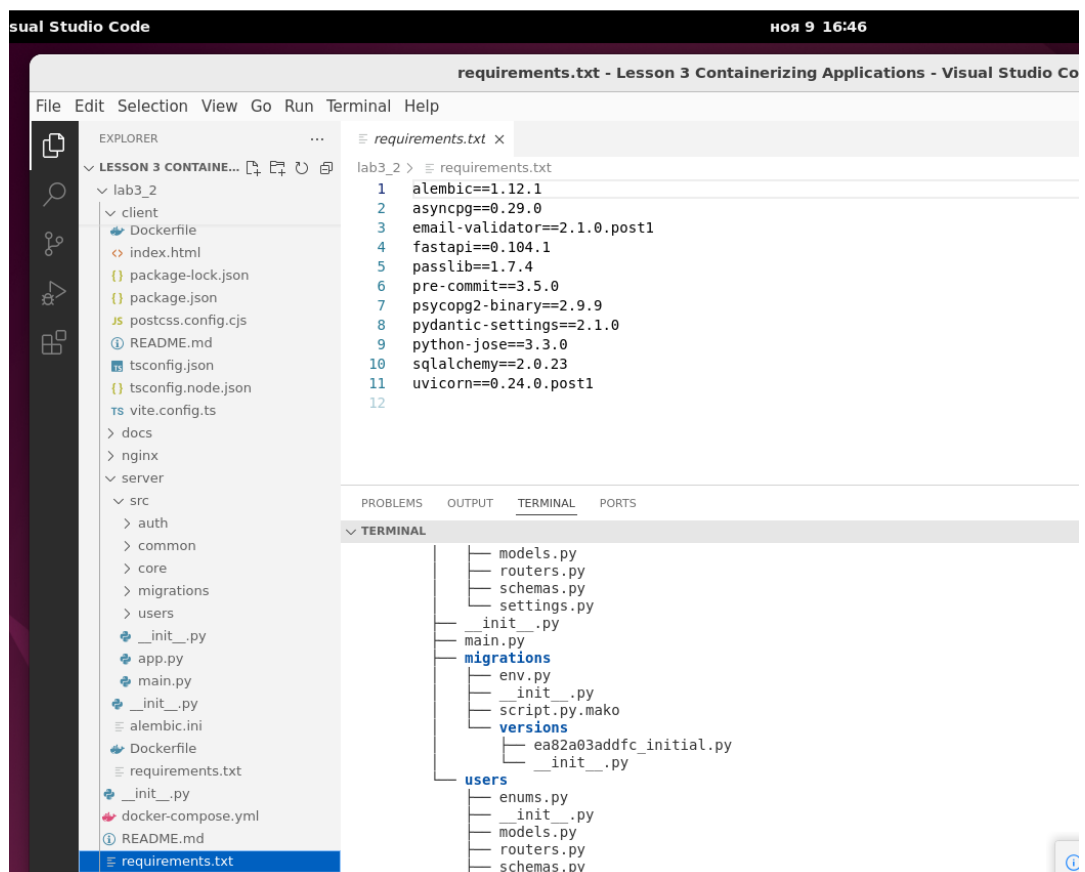
Проект по созданию сервиса базы данных на PostgreSQL

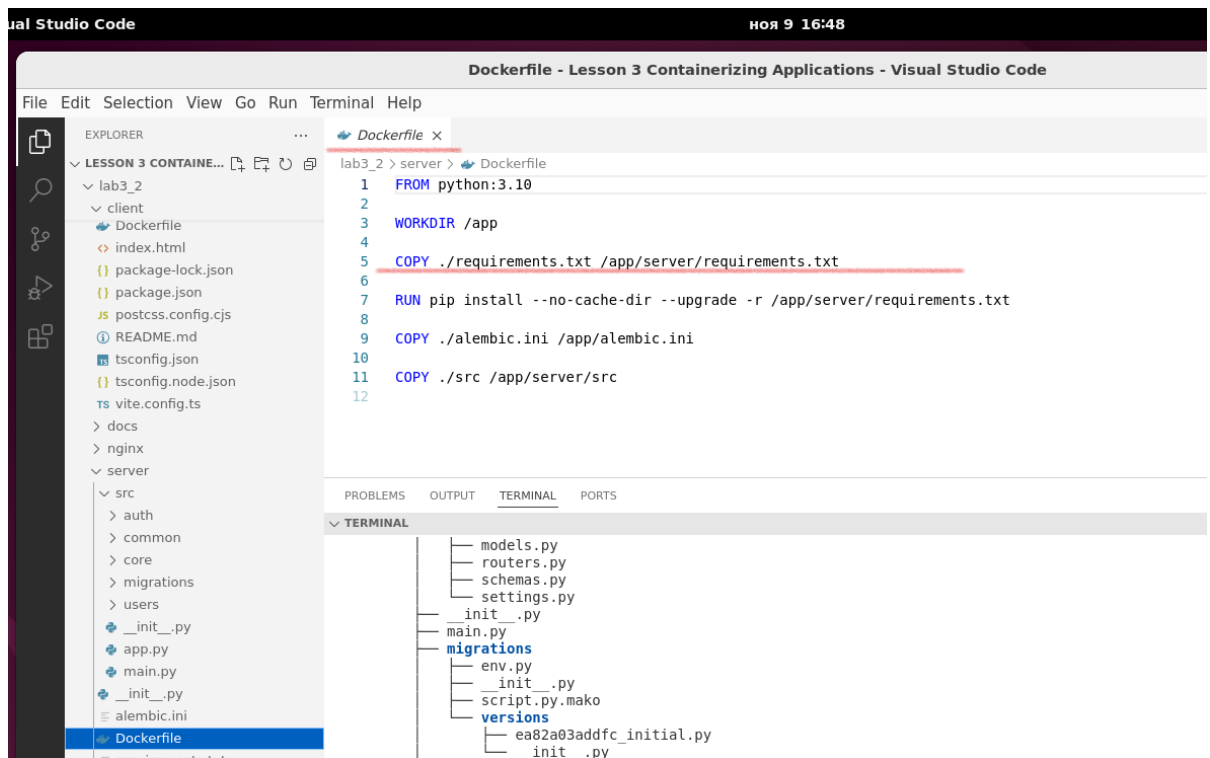
Проект содержит следующие основные компоненты:

- 1) **Dockerfile** - файл с последовательно выполняющимися инструкциями по созданию Docker. Инструкция содержит следующие команды:
  - **FROM (используемый язык, например, python)** - устанавливает базовый образ, от которого будет строиться новый образ.
  - **WORKDIR /app** - устанавливает рабочую директорию внутри контейнера на /app.
  - **COPY ./requirements.txt /app/server/requirements.txt \ ./alembic.ini /app/alembic.ini \ ./src /app/server/src** - копирует несколько файлов из контекста сборки (директории, где находится Dockerfile) в указанные пути внутри контейнера:
    - o **requirements.txt** - файл с перечнем зависимостей Python, которые нужно установить.
    - o **alembic.ini** - конфигурационный файл для Alembic (инструмента для миграций баз данных).
    - o **src** - директория, содержащая исходный код приложения.
  - **RUN pip install --no-cache-dir --upgrade -r /app/server/requirements.txt** - устанавливает зависимости Python, перечисленные в файле requirements.txt, используя pip. Флаг --no-cache-dir предотвращает использование кэша пакетов, а --upgrade обновляет пакеты до последней версии.
  - **CMD ["uvicorn", "server.src.main:app", "--host", "0.0.0.0", "--port", "8000"]** - запускает сервер uvicorn с приложением, определенным в модуле main из пакета server.src, на хосте "0.0.0.0" и порту "8000".
  - **VOLUME /app/data** - Все данные, которые будут записаны в этот каталог внутри контейнера, будут сохранены за пределами контейнера, что позволит сохранить их даже после остановки или удаления контейнера.
- 2) **docker-compose** — это инструмент для определения и запуска многоконтейнерных Docker-приложений. С помощью файла docker-compose.yml можно описать конфигурацию всех сервисов, сетей и томов, необходимых для вашего приложения, и запустить их одной командой. Это упрощает управление сложными приложениями, состоящими из нескольких взаимосвязанных сервисов. Например, веб-приложение может состоять из веб-сервера, базы данных и кэш-сервиса, и все эти компоненты можно легко управлять с помощью docker-compose.



- 3) **requirements** – текстовый файл, который содержит все библиотеки и их версии, необходимые для работы приложения. Обращение к requirements происходит из dockerfile





- 4) **alembic.ini** – журнал записей взаимодействия с внешней СУБД и в данном случае Python. Входит в раздел (каталог) source
- 5) **.env** – файл, в котором содержатся логин, пароль для авторизации в БД, например. Используется, чтобы эта информация не была раскрыта. Обращение к .env происходит из docker-compose, если в нем прописывается сервис по БД, или из dockerfile, если нет docker-compose и обращение к сервису БД идет через dockerfile. Или если создан отдельный файл для обращения к БД, то в нем.
- 6)

## 2. Конфигурационные файлы:

- Включите код Dockerfile, docker-compose.yml, и .env. Убедитесь, что они соответствуют требованиям задания.

Содержание **.env**:

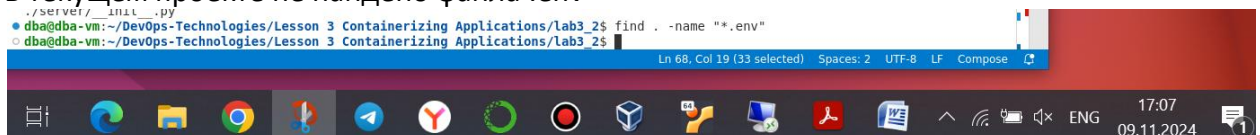
plaintext

APP\_PORT=8080

DB\_USER=user

DB\_PASSWORD=pass

В текущем проекте не найдено файла .env

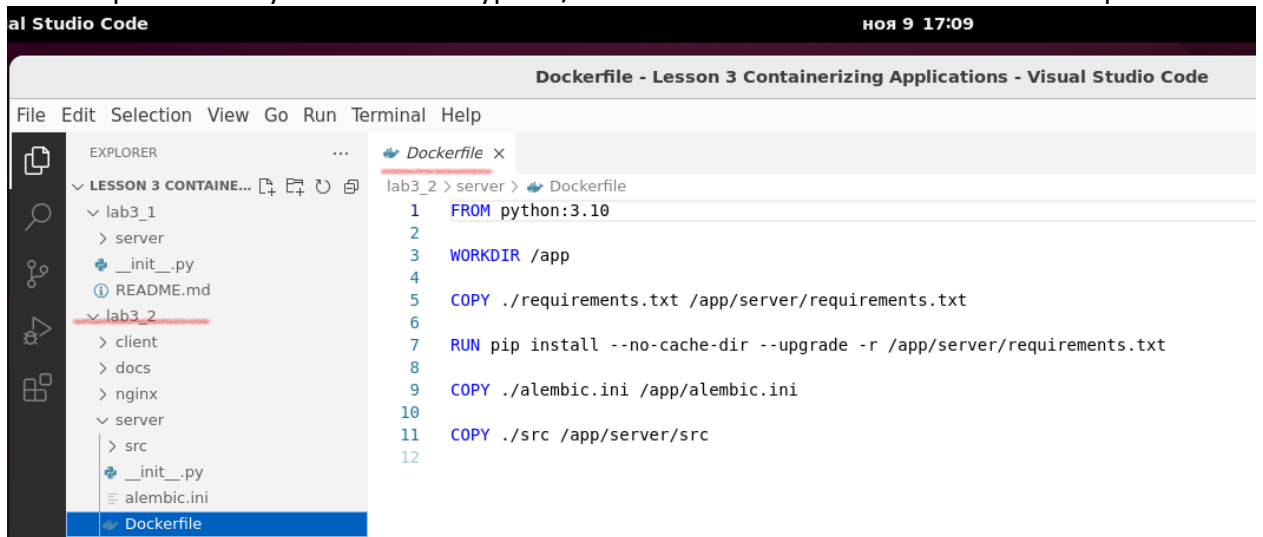




## Dockerfile

Прописан верно:

- указана версия языка
- все шаги ссылаются на отдельные файлы, а не прописаны в самом dockerfile
- было требование установить entrypoint/ command – Реализовано в docker-compose



The screenshot shows the Visual Studio Code interface with a Dockerfile open. The Explorer sidebar on the left shows a project structure with folders 'lab3\_1' and 'lab3\_2'. 'lab3\_2' contains subfolders 'client', 'docs', 'nginx', and 'server'. The 'server' folder contains files like 'src', '\_\_init\_\_.py', and 'alembic.ini'. The Dockerfile is located in the 'server' folder. The code in the Dockerfile is as follows:

```
1 FROM python:3.10
2
3 WORKDIR /app
4
5 COPY ./requirements.txt /app/server/requirements.txt
6
7 RUN pip install --no-cache-dir --upgrade -r /app/server/requirements.txt
8
9 COPY ./alembic.ini /app/alembic.ini
10
11 COPY ./src /app/server/src
12
```

## Docker-compose:



The screenshot shows the Visual Studio Code interface with a docker-compose.yml file open. The Explorer sidebar on the left shows the same project structure as the previous screenshot. The docker-compose.yml file is located in the 'server' folder. The code in the docker-compose.yml file is as follows:

```
1 version: '3'
2
3 services:
4   init:
5     container_name: init
6     build:
7       context: ./server
8       dockerfile: Dockerfile
9     env_file: .env
10    environment:
11      - PYTHONUNBUFFERED=0
12      - PYTHONPATH=/app/server
13      - ENVIRONMENT=production
14    command:
15      - sh
16      - -c
17      - 'alembic upgrade head'
18    depends_on:
19      - db
```

```
ноя 9 17:26
docker-compose.yml - Lesson 3 Containerizing Applications - Visual Studio Code
Terminal Help
lab3_2 > docker-compose.yml
3 services:
20
21 server:
22   container_name: server
23   build:
24     context: ./server
25     dockerfile: Dockerfile
26   env_file: .env
27   environment:
28     - PYTHONUNBUFFERED=0
29     - PYTHONPATH=/app/server
30     - ENVIRONMENT=production
31   restart: always
32   command:
33     - sh
34     - -c
35     - 'uvicorn server.src.main:app --host 0.0.0.0 --port ${SERVER_PORT:-8000}'
36   ports:
37     - '8000:${SERVER_PORT:-8000}'
38   volumes:
39     - ./server:/app/server
40   depends_on:
41     - db
42     - init
43   healthcheck:
44     test: ["CMD-SHELL", "curl -f http://localhost:${SERVER_PORT:-8000}/health || exit 1"]
```

Требование для docker-compose:

- 1) Автоматическая сборка образа из Dockerfile, лежащего в корневой папке проекта.- реализовано

у рабочему столу

```
ноя 9 17:29
docker-compose.yml - Lesson 3 Containerizing Applications - Visual Studio Code
Go Run Terminal Help
lab3_2 > docker-compose.yml
1 version: '3'
2
3 services:
4   init:
5     container_name: init
6     build:
7       context: ./server
8       dockerfile: Dockerfile
9     env_file: .env
10    environment:
11      - PYTHONUNBUFFERED=0
12      - PYTHONPATH=/app/server
13      - ENVIRONMENT=production
14    command:
15      - sh
16      - -c
17      - 'alembic upgrade head'
18    depends_on:
19      - db
20
21  server:
22    container_name: server
23    build:
24      context: ./server
25      dockerfile: Dockerfile
26    env_file: .env
27    environment:
28      - PYTHONUNBUFFERED=0
29      - PYTHONPATH=/app/server
30      - ENVIRONMENT=production
```

- 2) Присвоить имя каждому образу, созданному из Dockerfile  
Нет

### 3. Пояснение настроек Docker compose.

Объясните, как реализованы основные требования:

- описание каждого сервиса и его назначение
- как реализовано жесткое наименование контейнеров
- как применены depends on, volume, рикидка порта наружу, command //  
entrypoint
- описание healthcheck и сети

server:

container\_name: server --- жесткое именование контейнера

build:

context: ./server

dockerfile: Dockerfile

env\_file: .env

environment:

- PYTHONUNBUFFERED=0
- PYTHONPATH=/app/server
- ENVIRONMENT=production

restart: always

command:

- sh
- -c
- 'uvicorn server.src.main:app --host 0.0.0.0 --port \${SERVER\_PORT:-8000}'

ports:

- '8000:\${SERVER\_PORT:-8000}'

volumes: ---куда будут сохранены данные за пределами контейнера

- ./server:/app/server

depends\_on: ---- выолнение блока после блоков db, init

- db
- init

healthcheck: ---проверка корректности работы приложения

test: ["CMD-SHELL", "curl -f http://localhost:\${SERVER\_PORT:-8000}/health ||  
exit 1"] ---endpoint для проверки

interval: 30s --- с каким интервалом проверять

timeout: 10s ---сколько времени ждать ответ приложения

retries: 3 ---сколько попыток подряд совершить

networks:

- my\_network

Сети: настроены мосты

```
Terminal Help
Dockerfile ● docker-compose.yml ×
lab3_2 > docker-compose.yml
3  services:
21  server:
40      depends_on:
41          - db
42          - init
43      healthcheck:
44          test: ["CMD-SHELL", "curl -f http://loca
45          interval: 30s
46          timeout: 10s
47          retries: 3
48      networks:
49          - my_network
50
51      db:
52          container_name: db
53          image: postgres
54          restart: always
55          ports:
56          - '5432:5432'
57          env_file: .env
58          healthcheck:
59              test: ["CMD-SHELL", "pg_isready -U ${POS
60              interval: 30s
61              timeout: 10s
62              retries: 5
63          networks:
64              - my_network
65
66      networks:
67      my_network:
68      driver: bridge
```

#### 4. Процесс запуска.

- Приведите команды для сборки и запуска проекта. Опишите, что происходит на каждом этапе запуска, например, порядок запуска сервисов и их проверка.

- 1) находясь в главной директории запускаемого проекта, выполнить команду сборки образа  
**sudo docker compose up -d**
- 2) проверить ответ главной точки входа в приложение  
**curl <http://localhost:8000>**
- 3) в текущем проекте порядок запуска: **db, init, server**

#### 5. Результат работы.

- Докажите, что проект работает корректно. Сделайте скриншоты с запущенными контейнерами (docker ps) и работающими сервисами (например, вывод команд или доступ к приложению через браузер).
- Приведите результаты тестирования healthcheck и других проверок, если

После запуска формирования образов вышли 2 ошибки:

- непонятя строки version:3 в docker-compose
- отсутствует файл .env -

#### ВОЗМОЖНО.

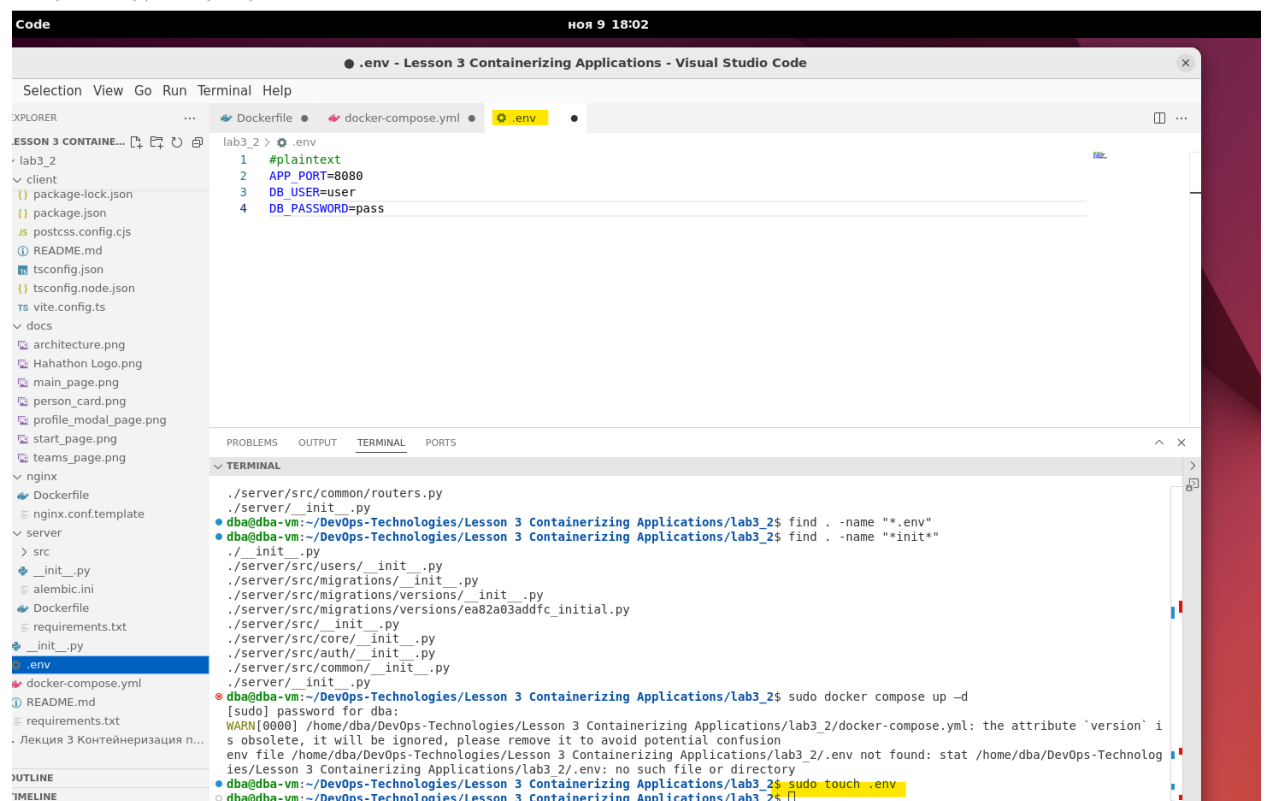
```
dba@dba-vm:~/DevOps-Technologies/Lesson 3 Containerizing Applications/lab3_2$ sudo docker compose up -d
[sudo] password for dba:
WARN[0000] /home/dba/DevOps-Technologies/Lesson 3 Containerizing Applications/lab3_2/docker-compose.yml: the attribute `version` i
s obsolete, it will be ignored, please remove it to avoid potential confusion
env file /home/dba/DevOps-Technologies/Lesson 3 Containerizing Applications/lab3_2/.env not found: stat /home/dba/DevOps-Technolog
ies/Lesson 3 Containerizing Applications/lab3_2/.env: no such file or directory
dba@dba-vm:~/DevOps-Technologies/Lesson 3 Containerizing Applications/lab3_2$
```

#### Закомментировала строку

почему столу

```
ноя 9 18:00
● docker-compose.yml - Lesson 3 Containerizing Applications - Visual Studio Code
Run Terminal Help
... Dockerfile ● docker-compose.yml ●
PPLICA... lab3_2 > docker-compose.yml
1  #version: '3'
2
3  services:
4    init:
5      container_name: init
6      build:
7        context: ./server
8      dockerfile: Dockerfile
```

## Создала файл .env



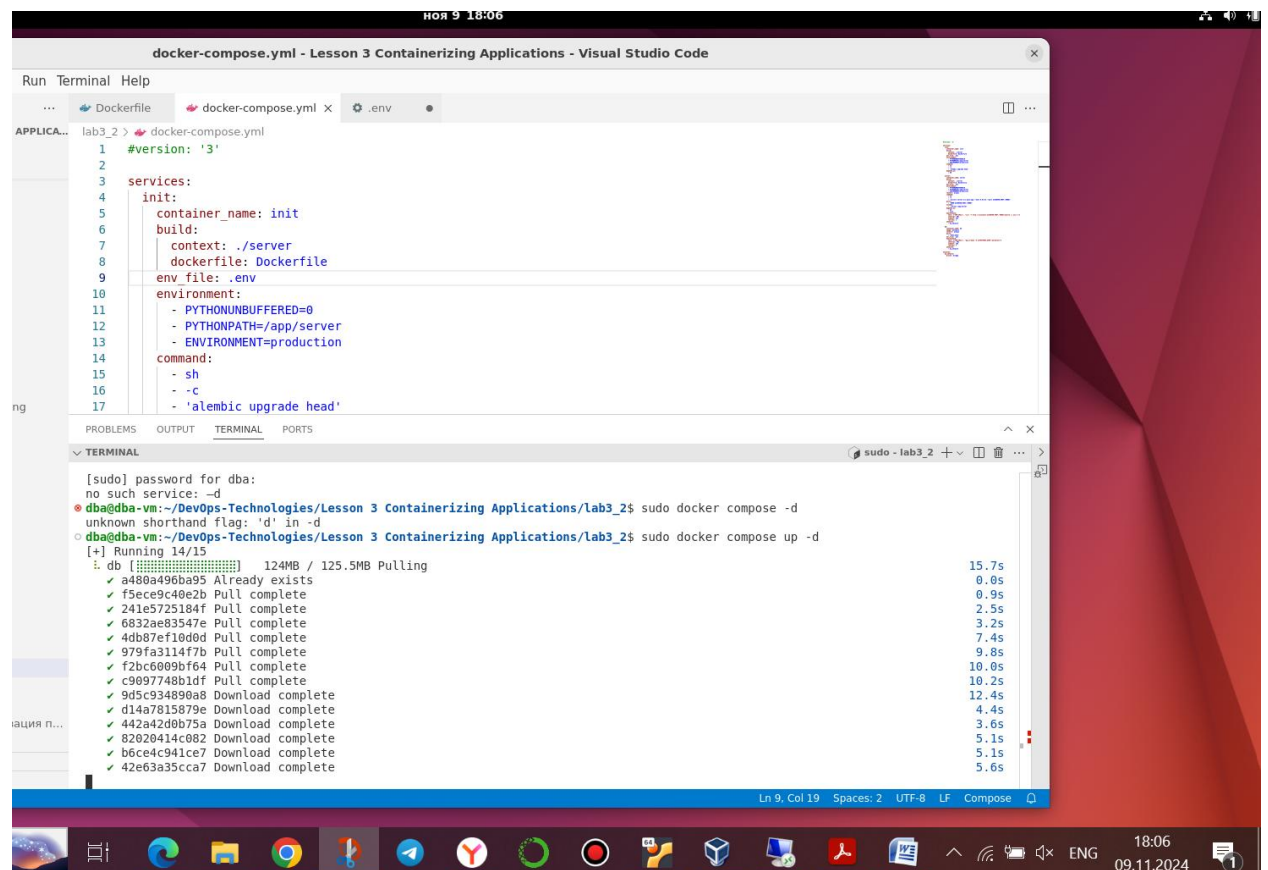
The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure. The main editor window shows the `.env` file with the following content:

```
1 #plaintext
2 APP_PORT=8080
3 DB_USER=user
4 DB_PASSWORD=pass
```

The terminal at the bottom shows the following commands and output:

```
./server/src/common/routers.py
./server/src/initial.py
dba@dba-vm:~/DevOps-Technologies/Lesson 3 Containerizing Applications/Lab3_2$ find . -name "*.env"
dba@dba-vm:~/DevOps-Technologies/Lesson 3 Containerizing Applications/Lab3_2$ find . -name "*init*"
./init.py
./server/src/users/_init_.py
./server/src/migrations/_init_.py
./server/src/migrations/versions/_init_.py
./server/src/migrations/versions/ea82a03addfc_initial.py
./server/src/_init_.py
./server/src/core/_init_.py
./server/src/auth/_init_.py
./server/src/common/_init_.py
./server/_init_.py
dba@dba-vm:~/DevOps-Technologies/Lesson 3 Containerizing Applications/Lab3_2$ sudo docker compose up -d
[sudo] password for dba:
WARN[0000] /home/dba/DevOps-Technologies/Lesson 3 Containerizing Applications/Lab3_2/docker-compose.yml: the attribute 'version' is
s obsolete, it will be ignored, please remove it to avoid potential confusion
env file /home/dba/DevOps-Technologies/Lesson 3 Containerizing Applications/Lab3_2/.env not found: stat /home/dba/DevOps-Techno
gies/Lesson 3 Containerizing Applications/Lab3_2/.env: no such file or directory
dba@dba-vm:~/DevOps-Technologies/Lesson 3 Containerizing Applications/Lab3_2$ sudo touch .env
dba@dba-vm:~/DevOps-Technologies/Lesson 3 Containerizing Applications/Lab3_2$
```

## После этих изменений были сформированы образы



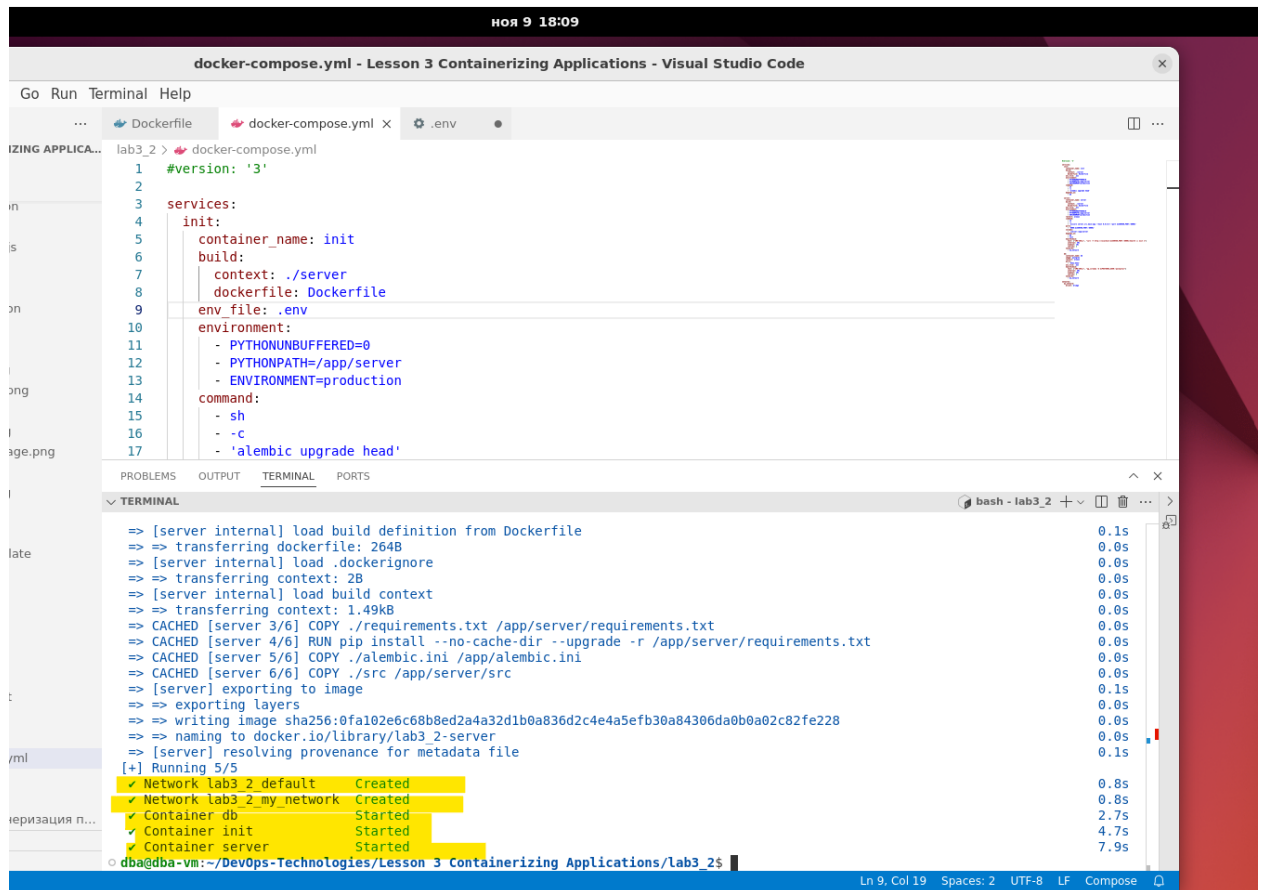
The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure. The main editor window shows the `docker-compose.yml` file with the following content:

```
1 #version: '3'
2
3 services:
4   init:
5     container_name: init
6     build:
7       context: ./server
8       dockerfile: Dockerfile
9     env_file: .env
10    environment:
11      - PYTHONUNBUFFERED=0
12      - PYTHONPATH=app/server
13      - ENVIRONMENT=production
14    command:
15      - sh
16      - -c
17      - 'alembic upgrade head'
```

The terminal at the bottom shows the following commands and output:

```
[sudo] password for dba:
no such service: -d
dba@dba-vm:~/DevOps-Technologies/Lesson 3 Containerizing Applications/Lab3_2$ sudo docker compose -d
unknown shorthand flag: 'd' in -d
dba@dba-vm:~/DevOps-Technologies/Lesson 3 Containerizing Applications/Lab3_2$ sudo docker compose up -d
[+] Running 14/15
  db [-----] 124MB / 125.5MB Pulling 15.7s
  ✓ a480a496ba95 Already exists 0.0s
  ✓ f5ce9c40e2b Pull complete 0.9s
  ✓ 241e5725184f Pull complete 2.5s
  ✓ 6832ae03547e Pull complete 3.2s
  ✓ 4db07ef10d0d Pull complete 7.4s
  ✓ 979fa3114f7b Pull complete 9.8s
  ✓ f2bc6009bf64 Pull complete 10.0s
  ✓ c9097748b1df Pull complete 10.2s
  ✓ 9d5c934890a8 Download complete 12.4s
  ✓ d14a7815879e Download complete 4.4s
  ✓ 442a42d0b75a Download complete 3.6s
  ✓ 82020414c082 Download complete 5.1s
  ✓ b6ce4c941ce7 Download complete 5.1s
  ✓ 42e63a35cca7 Download complete 5.6s
```

му рабочему столу



The screenshot shows a Visual Studio Code window with a file named `docker-compose.yml` open. The file contains a Docker Compose configuration for a service named `init`. The configuration includes a build context, environment variables, and a command to run `sh` and `c` (likely `alembic upgrade head`). The terminal output shows the process of building the image, including downloading the Dockerfile, context, and build context, and then running the command. The output also shows the creation of the `lab3_2` network and the starting of the `db` and `init` containers.

```
1 #version: '3'
2
3 services:
4   init:
5     container_name: init
6     build:
7       context: ../server
8       dockerfile: Dockerfile
9     env_file: .env
10    environment:
11      - PYTHONUNBUFFERED=0
12      - PYTHONPATH=/app/server
13      - ENVIRONMENT=production
14    command:
15      - sh
16      - c
17      - 'alembic upgrade head'
```

```
=> [server internal] load build definition from Dockerfile
=> transferring dockerfile: 264B
=> [server internal] load .dockerignore
=> transferring context: 2B
=> [server internal] load build context
=> transferring context: 1.49kB
=> CACHED [server 3/6] COPY ./requirements.txt /app/server/requirements.txt
=> CACHED [server 4/6] RUN pip install --no-cache-dir --upgrade -r /app/server/requirements.txt
=> CACHED [server 5/6] COPY ./alembic.ini /app/alembic.ini
=> CACHED [server 6/6] COPY ./src /app/server/src
=> [server] exporting to image
=> exporting layers
=> writing image sha256:0fa102e6c68b8ed2a4a32d1b0a836d2c4e4a5efb30a84306da0b0a02c82fe228
=> naming to docker.io/library/lab3_2-server
=> [server] resolving provenance for metadata file
[+] Running 5/5
 ✓ Network lab3_2 default Created
 ✓ Network lab3_2 my network Created
 ✓ Container db Started
 ✓ Container init Started
 ✓ Container server Started
```

## Запущенные контейнеры `sudo docker ps`

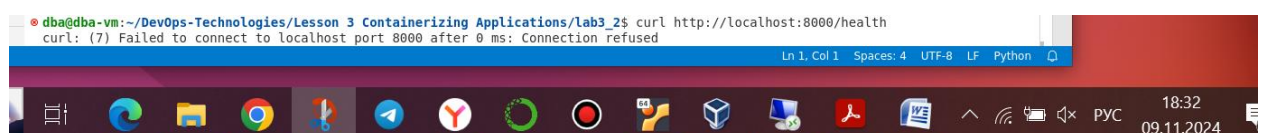


The screenshot shows the output of the `sudo docker ps` command. It lists the running containers, including the `db` and `server` containers. The output shows the container ID, image, command, created time, status, ports, and names.

```
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
94c755f44daf   lab3_2-server  "sh -c 'uvicorn serv..." About a minute ago  Restarting (1) 3 seconds ago  server
94cade61e2d3   postgres      "docker-entrypoint.s..." About a minute ago  Restarting (1) 17 seconds ago  db
```

К сожалению , проверить healthcheck не удалось: при первой попытке вышло сообщение ***curl: (7) Failed to connect to localhost port 8000 after 0 ms: Connection refused*** после чего остановила и удалила контейнеры, удалила образы и запустила формирование образа повторно. При повторной проверке , вышла та же ошибка ***curl (7)***

```
curl http://localhost:8000/health
curl http://localhost:8000/api/docs
sudo docker stop $(sudo docker ps -aq) && sudo docker rm $(sudo docker ps -aq)
sudo docker rmi $(sudo docker images -q)
sudo docker system prune -a
sudo docker compose up -d
curl http://localhost:8000/health
```



The screenshot shows the terminal output of the `curl http://localhost:8000/health` command. The output is the same error message as before: ***curl: (7) Failed to connect to localhost port 8000 after 0 ms: Connection refused***.

```
curl: (7) Failed to connect to localhost port 8000 after 0 ms: Connection refused
```

Также потом изменила данные в файле .env, но это тоже не привело к ответу сайта

