Homework_Lesson22_Report

Цель: получить практический опыт написания Dockerfile, развертывания приложений с использованием Docker-compose

Задание 1: Создание Dockerfile для приложения веб-сервера. Вам необходимо написать Dockerfile для создания контейнера с приложением веб-сервера на основе образа Ubuntu 20.04. Приложение должно быть запущено на порту 8080 и должно отдавать статические файлы из каталога /app/static.

Задание 2 – развертывание приложения с помощью Docker-compose

Задание 1

Задание 1: Создание Dockerfile для приложения веб-сервера. Вам необходимо написать Dockerfile для создания контейнера с приложением веб-сервера на основе образа Ubuntu 20.04. Приложение должно быть запущено на порту 8080 и должно отдавать статические файлы из каталога /app/static.

1. Создайте новый файл Dockerfile в пустой директории на вашем локальном компьютере.
2. Напишите инструкцию FROM, которая указывает базовый образ Ubuntu 20.04.
3. Установите необходимые зависимости с помощью инструкции RUN. Установите пакеты nginx и curl, а также создайте каталог /app/static.
4. Скопируйте файл конфигурации nginx из вашего локального каталога внутрь контейнера с помощью инструкции COPY.
5. Скопируйте статические файлы из каталога /app/static на вашем локальном компьютере внутрь контейнера с помощью инструкции COPY.
6. Используйте инструкцию EXPOSE для открытия порта 8080.
7. Используйте инструкцию CMD для запуска команды nginx с указанием пути к файлу конфигурации, который вы скопировали на шаге 4.
8. Сохраните файл Dockerfile и соберите образ с помощью команды docker build.
9. Запустите контейнер из образа с помощью команды docker run и проверьте, что веб-сервер отдает статические файлы из каталога /app/static на порту 8080.

Напишем докер файл для нашего веб-сервера.

FROM ubuntu:20.04 # какой образ берем за основу

ENV DEBIAN_FRONTEND=noninteractive # включаем не интерактивный режим apt

RUN apt update && apt install -y nginx curl && \ # устанавливаем nginx curl
mkdir -p /app/static # создаем директорию /app/static

COPY nginx.conf /etc/nginx/nginx.conf #копируем конфиг /nginx

COPY ./app/static /app/static # копируем статические файлы index.html и 1.jpg

EXPOSE 8080 # открываем порт 8080

CMD ["nginx", "-g", "daemon off;"] #запускаем nginx на переднем плане, а не в фоновом режиме.

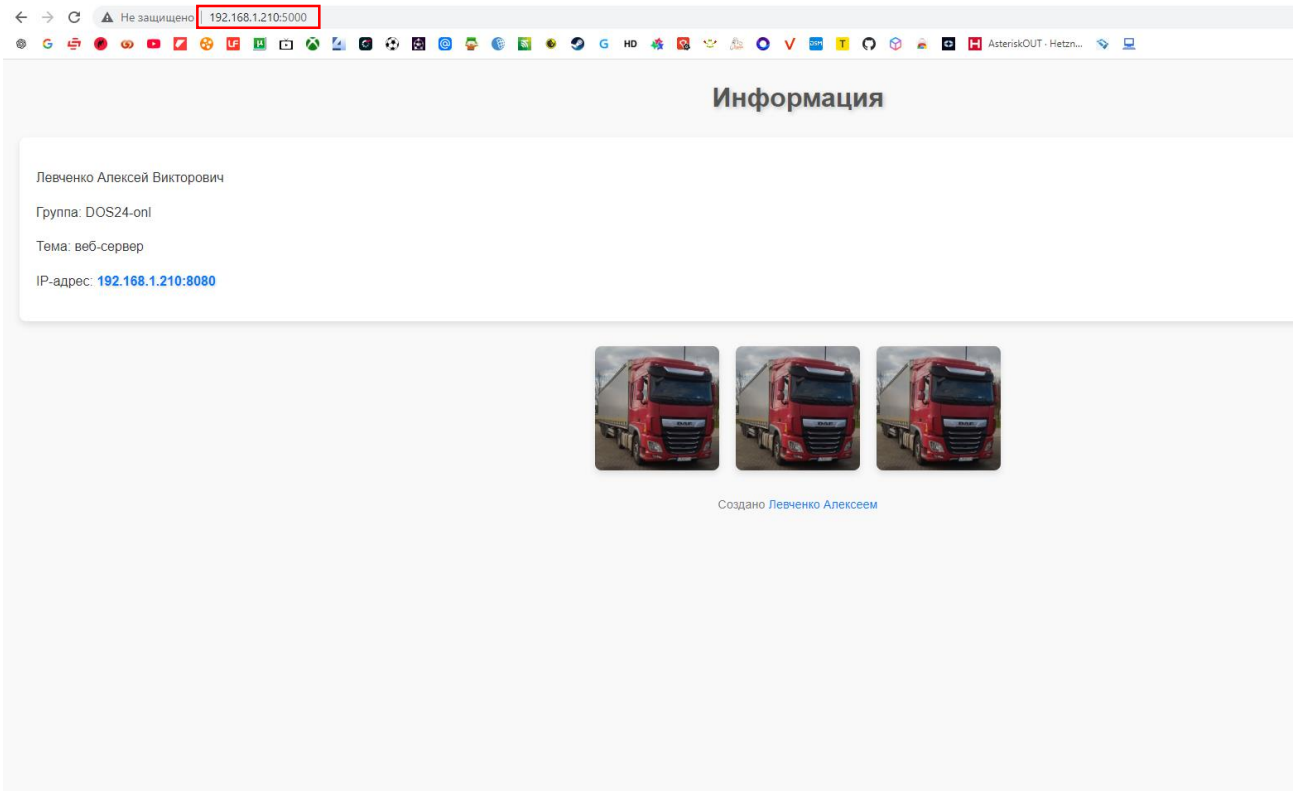
Сбилдим наш dockerfile.

```
avl@ubuntu-s24:~/docker/nginx$ sudo docker build -t web .
[+] Building 106.1s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 292B
 => [internal] load metadata for docker.io/library/ubuntu:20.04
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM docker.io/library/ubuntu:20.04@sha256:8e5c4f0285ecbb4ead070431d29b576a530d3166df73ec44affc1cd27555141b
 => => resolve docker.io/library/ubuntu:20.04@sha256:8e5c4f0285ecbb4ead070431d29b576a530d3166df73ec44affc1cd27555141b
 => => sha256:8e5c4f0285ecbb4ead070431d29b576a530d3166df73ec44affc1cd27555141b 6.69kB / 6.69kB
 => => sha256:e5a6aeef391a8a9bdaee3de6b28f393837c479d8217324a2340b64e45a81e0ef 424B / 424B
 => => sha256:6013ae1a63c2ee58a8949f03c6366a3ef6a2f386a7db27d86de2de965e9f450b 2.30kB / 2.30kB
 => => sha256:d9802f032d6798e2086607424bfe88cb8ec1d6f116e11cd99592dcaf261e9cd2 27.51MB / 27.51MB
 => => extracting sha256:d9802f032d6798e2086607424bfe88cb8ec1d6f116e11cd99592dcaf261e9cd2
 => [internal] load build context
 => => transferring context: 149.69kB
 => [2/4] RUN apt-get update && apt-get install -y nginx curl && mkdir -p /app/static
 => [3/4] COPY nginx.conf /etc/nginx/nginx.conf
 => [4/4] COPY ./app/static /app/static
 => exporting to image
 => => exporting layers
 => => writing image sha256:e8d4272db8c06375a380bdf7a76705bef8b9df00ef23d5d224a646afb53b5074
 => => naming to docker.io/library/web
avl@ubuntu-s24:~/docker/nginx$
```

Запустим контейнер с нашем image. Дадим ему имя, перенаправим порты и запустим контейнер в отсоединённом режиме. Далее посмотрим за запущенные контейнеры. Как видим контейнер наш работает.

```
avl@ubuntu-s24:~/docker/nginx$ sudo docker run --name web -p 5000:8080 -d web
996c0bcc408fe9cf19aedebb19563e3339e3a38af77619f00c8d842f0d780dc3
avl@ubuntu-s24:~/docker/nginx$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
996c0bcc408f   web       "nginx -g 'daemon of…"   6 seconds ago   Up 6 seconds   0.0.0.0:5000->8080/tcp, [::]:5000->8080/tcp
avl@ubuntu-s24:~/docker/nginx$
```

Войдем в браузер и посмотри и перейдем по ip 192.168.1.210:5000 это ip хоста на котором работает контейнер и проброшенный порт. Как видим наша html отображается корректно.


Не защищено
192.168.1.210:5000
AsteriskOUT - Hetzn...
Информация
Левченко Алексей Викторович
Группа: DOS24-onl
Тема: веб-сервер
IP-адрес: 192.168.1.210:8080
Создано Левченко Алексеем

Задание 2

Задание 2 – развертывание приложения с помощью Docker-compose
Шаги, которые необходимо выполнить:
1. Создайте новый файл docker-compose.yml в пустой директории на вашем локальном компьютере.
2. Напишите инструкцию version в версии 3.
3. Определите сервис для базы данных PostgreSQL. Назовите его "db". Используйте образ postgres:latest, задайте переменные окружения POSTGRES_USER, POSTGRES_PASSWORD и POSTGRES_DB для установки пользовательского имени, пароля и имени базы данных соответственно.
4. Определите сервис для веб-сервера на основе образа NGINX. Назовите его "web". Используйте образ nginx:latest. Определите порт, на котором должен работать сервер, с помощью инструкции ports. Задайте путь к файлам конфигурации NGINX внутри контейнера, используя инструкцию volumes.
5.* Определите ссылку на сервис базы данных в сервисе веб-сервера. Используйте инструкцию links.
6. Сохраните файл docker-compose.yml и запустите приложение с помощью команды docker-compose up.
7. Проверьте, что приложение работает, перейдя в браузере на localhost:80.

Пишем файл docker-compose.yml исходя из задания. Веб сервер у нас будет web1. В мы будем его билдить из докерфайла. Внешний порт хоста 5001.

```
 GNU nano 7.2
version: "3"

services:
  db:
    image: postgres:latest
    environment:
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: mypassword
      POSTGRES_DB: mydatabase
    volumes:
      - db_data:/var/lib/postgresql/data

  web1:
    build:
      context: .
    ports:
      - "5001:80"
    depends_on:
      - db
    links:
      - db
volumes:
  db_data:
```

Doskerfile nginx. Конфиг nginx и статические файлы копируем из прошлого задания.

```
FROM nginx

COPY nginx.conf /etc/nginx/nginx.conf

COPY ./app/static /app/static

EXPOSE 80
```

```
avl@ubuntu-s24:~/docker/compose_nginx$ sudo docker-compose up
Creating network "compose_nginx_default" with the default driver
Creating volume "compose_nginx_db_data" with default driver
Pulling db (postgres:latest)...
latest: Pulling from library/postgres
af302e5c37e9: Pull complete
23db180a1f67: Pull complete
dc59dd9c8eb3: Pull complete
aec09e638045: Pull complete
4dd47a683737: Pull complete
7cebbe7849b3: Pull complete
dc4330b02129: Pull complete
498cc40b9fe9: Pull complete
6d3411bb4696: Pull complete
8f14f34d54d3: Pull complete
88d4f7416643: Pull complete
e91ad5cfb8d0: Pull complete
e0c4d5055fb9: Pull complete
254ee626d709: Pull complete
Digest: sha256:87ec5e0a167dc7d4831729f9e1d2ee7b8597dcc49ccd9e43cc5f89e808d2adae
Status: Downloaded newer image for postgres:latest
Building web1
[+] Building 15.9s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 132B
 => [internal] load metadata for docker.io/library/nginx:latest
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/3] FROM docker.io/library/nginx:latest@sha256:0a399eb16751829e1af26fea27b20c3ec28d7ab1fb72182879dcae1cca21206a
 => => resolve docker.io/library/nginx:latest@sha256:0a399eb16751829e1af26fea27b20c3ec28d7ab1fb72182879dcae1cca21206a
 => => sha256:0a399eb16751829e1af26fea27b20c3ec28d7ab1fb72182879dcae1cca21206a 10.27kB / 10.27kB
 => => sha256:2426c815287ed75a3a33dd28512eba4f0f783946844209ccf3fa8990817a4eb9 2.29kB / 2.29kB
 => => sha256:9bea9f2796e236cb18c2b3ad561ff29f655d1001f9ec7247a0bc5e08d25652a1 8.58kB / 8.58kB
 => => sha256:841e383b441eda86c164d8e833da080b951c951d612853cac2fd44eed0d72226 627B / 627B
 => => sha256:0256c04a8d84ab5ae434b607055df8fdfe0278dba43f3e603a861dc7e26da395 957B / 957B
 => => sha256:207b812743af01771d6c94b3ebc59543bf3c6fefe5d4d9ebfa6e5e5558801d34 43.84MB / 43.84MB
 => => sha256:38e992d287c563d332ed8c9bdefec32a84f50f94cc39bb4729a3e73cefde88eb 406B / 406B
 => => sha256:9e9aab598f58e86706b73c23e0ff33fc4eb60a5e69424147f9cb7c02e853323b 1.21kB / 1.21kB
 => => sha256:4de87b37f4ad0b499be3db4cc6831db3d8158867f12dabc0bf42046b56784d3b 1.40kB / 1.40kB
 => => extracting sha256:207b812743af01771d6c94b3ebc59543bf3c6fefe5d4d9ebfa6e5e5558801d34
 => => extracting sha256:841e383b441eda86c164d8e833da080b951c951d612853cac2fd44eed0d72226
 => => extracting sha256:0256c04a8d84ab5ae434b607055df8fdfe0278dba43f3e603a861dc7e26da395
 => => extracting sha256:38e992d287c563d332ed8c9bdefec32a84f50f94cc39bb4729a3e73cefde88eb
 => => extracting sha256:9e9aab598f58e86706b73c23e0ff33fc4eb60a5e69424147f9cb7c02e853323b
 => => extracting sha256:4de87b37f4ad0b499be3db4cc6831db3d8158867f12dabc0bf42046b56784d3b
 => [internal] load build context
 => => transferring context: 150.10kB
 => [2/3] COPY nginx.conf /etc/nginx/nginx.conf
 => [3/3] COPY ./app/static /app/static
 => exporting to image
 => => exporting layers
 => => writing image sha256:e76f6c3f4a5faabd913d9fcdd8cd29ae134aa83f7f410890f09e0ad026fb9917
 => => naming to docker.io/library/compose_nginx_web1
WARNING: Image for service web1 was built because it did not already exist. To rebuild this image you must use `docker-compose build` or `docker-compose up --build`.
Creating compose_nginx_db_1 ... done
Creating compose_nginx_web1_1 ... done
Attaching to compose_nginx_db_1, compose_nginx_web1_1
db_1    | The files belonging to this database system will be owned by user "postgres".
db_1    | This user must also own the server process.
```

Забыл в команде dosker-compose up указать –d. Грохнем контейнеры и запустим заново. Заметим что image для web1 сбилдился автоматически.

Запустим повторно dosker-compose up –d.

```
avl@ubuntu-s24:~/docker/compose_nginx$ sudo docker-compose up -d
Starting compose_nginx_db_1 ... done
Starting compose_nginx_web1_1 ... done
```

Проверим запущенные контейнеры.

```
avl@ubuntu-s24:~/docker/compose_nginx$ sudo docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED             STATUS             PORTS                                                  NAMES
0bbe428df9bc   compose_nginx_web1   "/docker-entrypoint.…"   About an hour ago   Up About an hour   0.0.0.0:5001->80/tcp, [::]:5001->80/tcp                compose_nginx_web1_1
f10aff563074   postgres:latest      "docker-entrypoint.s…"   About an hour ago   Up About an hour   5432/tcp                                               compose_nginx_db_1
c17989c2d26    web                  "nginx -g 'daemon of…"   21 hours ago        Up 21 hours        0.0.0.0:5000->8080/tcp, [::]:5000->8080/tcp            web
```

Контейнеры работают.

Проверим веб сервис. Как видим сервис работает.