Задание:

1. Установить Docker в виртуальную машину(или в VDS) – делаю по инструкции [Установка и использование Docker в Ubuntu 20.04]

Обновляем существующий список пакетов:
```
sudo apt update
```

Устанавливаем пакеты, которые позволяют apt использовать пакеты через HTTPS:
```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

Добавляем ключ GPG для официального репозитория Docker:
```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Добавляем репозиторий Docker в источники APT:
```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

Обновляем базу данных пакетов из добавленного репозитория:
```
sudo apt update
```

Проверяем установку:
```
apt-cache policy docker-ce
```
Вывод:
```
docker-ce:
  Установлен: (отсутствует)
  Кандидат:   5:20.10.18~3-0~ubuntu-focal
```

Установим Docker:
```
sudo apt install docker-ce
```

Проверим, что он запущен:
```
sudo systemctl status docker
```
- Ок.

Проверим установку контейнеров:
```
docker run hello-world
```
- контейнер установился
```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```
- Ок.

Устанавливаем docker compose по инструкции [Как установить Docker Compose на Ubuntu 18.04]

Проверяем текущую версию и обновляемся:
```
abubakirov@abubakirov-VirtualBox:~$ sudo curl -L https://github.com/docker/compose/releases/
download/v2.10.2/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
```
(обратить внимание на буковку v в номере версии!)

Даем разрешение на запуск:
```
sudo chmod +x /usr/local/bin/docker-compose
```

Проверяем: `sudo docker-compose -v` ответ: `Docker Compose version v2.10.2` - Ок.

Настроить набор контейнеров через docker compose по инструкции по ссылке: Установка WordPress с помощью Docker Compose.
Часть с настройкой certbot и HTTPS опустить, если у вас нет настоящего домена и белого IP.

Создаем в домашней папке директорию wordpress: `sudo mkdir wordpress && cd wordpress`
Создаем в этой папке папку с настройками nginx: `sudo mkdir nginx-conf`
Создаем файл настроек: `sudo nano nginx-conf/nginx.conf`

Вносим настройки:

```
server {
        listen 80;
        listen [::]:80;

        server_name test.loc www.test.loc;

        index index.php index.html index.htm;

        root /var/www/html;

        location ~ /.well-known/acme-challenge {
                allow all;
                root /var/www/html;
        }

        location / {
                try_files $uri $uri/ /index.php$is_args$args;
        }

        location ~ \.php$ {
                try_files $uri =404;
                fastcgi_split_path_info ^(.+\.php)(/.+)$;
                fastcgi_pass wordpress:9000;
                fastcgi_index index.php;
                include fastcgi_params;
                fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
                fastcgi_param PATH_INFO $fastcgi_path_info;
        }

        location ~ /\.ht {
                deny all;
        }

        location = /favicon.ico {
                log_not_found off; access_log off;
        }
        location = /robots.txt {
                log_not_found off; access_log off; allow all;
        }
        location ~* \.(css|gif|ico|jpeg|jpg|js|png)$ {
                expires max;
                log_not_found off;
        }
}
```

Создаем файл с настройками базы данных: `sudo nano .env`

```
MYSQL_ROOT_PASSWORD=test
MYSQL_USER=admin
MYSQL_PASSWORD=test
```

Вносим файл .env в файл с настройками игнора для docker: `sudo nano .dockerignore`

```
.env
.git
docker-compose.yml
.dockerignore
```

Создаем файл настроек для docker-compose:

`sudo nano docker-compose.yml`

Определяем в нем службы (настройки работы контейнеров):

```
version: '3'

services:
  db:
    image: mysql:8.0
    container_name: db
    restart: unless-stopped
    env_file: .env
    environment:
      - MYSQL_DATABASE=wordpress
    volumes:
      - dbdata:/var/lib/mysql
    command: '--default-authentication-plugin=mysql_native_password'
    networks:
     - app-network

  wordpress:
    depends_on:
      - db
    image: wordpress:5.1.1-fpm-alpine
    container_name: wordpress
    restart: unless-stopped
    env_file: .env
    environment:
      - WORDPRESS_DB_HOST=db:3306
      - WORDPRESS_DB_USER=$MYSQL_USER
      - WORDPRESS_DB_PASSWORD=$MYSQL_PASSWORD
      - WORDPRESS_DB_NAME=wordpress
    volumes:
      - wordpress:/var/www/html
    networks:
      - app-network

  webserver:
    depends_on:
      - wordpress
    image: nginx:1.15.12-alpine
    container_name: webserver
    restart: unless-stopped
    ports:
      - "80:80"
    volumes:
      - wordpress:/var/www/html
      - ./nginx-conf:/etc/nginx/conf.d
    networks:
      - app-network

volumes:
  wordpress:
  dbdata:

networks:
  app-network:
    driver: bridge
```

Запускаем контейнеры: `sudo docker-compose up -d`

Проверяем:

```
abubakirov@abubakirov-VirtualBox:~/wordpress$ sudo docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED              STATUS              PORTS                                   NA
MES
de4a0a868c9d   nginx:1.15.12-alpine       "nginx -g 'daemon of…"   About a minute ago   Up About a minute   0.0.0.0:80->80/tcp, :::80->80/tcp       we
bserver
c173ea333ce8   wordpress:5.1.1-fpm-alpine "docker-entrypoint.s…"   About a minute ago   Up About a minute   9000/tcp                                wo
rdpress
4e7c6658dd86   mysql:8.0                  "docker-entrypoint.s…"   About a minute ago   Up About a minute   3306/tcp, 33060/tcp                     db
```

Ок.

Далее, через браузер заходим по адресу 10.0.1.12/test.loc – видим приглашение wordpress, заканчиваем настройку.

2. * Запустить два контейнера, связанные одной сетью (используя документацию). Первый контейнер БД (например, образ mariadb:10.8), второй контейнер – phpmyadmin. Получить доступ к БД в первом контейнере через второй контейнер (веб-интерфейс phpmyadmin).

Создаем новую папку database и переходим в нее: `sudo mkdir database && cd database`

Создаем файл настроек для docker-compose: `sudo nano docker-compose.yml`

Определяем в нем службы (настройки работы контейнеров):

```
version: '3'

services:
  db:
    image: mariadb:10.8
    container_name: mariadb
    environment:
      MARIADB_ROOT_PASSWORD: my_secret_password
      MARIADB_DATABASE: app_db
      MARIADB_USER: db_user
      MARIADB_PASSWORD: db_user_pass
    ports:
      - "6033:3306"
    volumes:
      - dbdata:/var/lib/mariadb

  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: pma
    links:
      - db
    environment:
      PMA_HOST: mariadb
      PMA_PORT: 3306
      PMA_ARBITRARY: 1
    restart: always
    ports:
      - 8081:80

volumes:
  dbdata:
```

Запускаем контейнеры: `sudo docker-compose up -d`

Проверяем:

```
abubakirov@abubakirov-VirtualBox:~/database$ sudo docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS         PORTS                                          NAMES
828c935eb432   phpmyadmin/phpmyadmin  "/docker-entrypoint.…"   24 seconds ago  Up 20 seconds  0.0.0.0:8081->80/tcp, :::8081->80/tcp          pma
a662e8fe2445   mariadb:10.8           "docker-entrypoint.s…"   25 seconds ago  Up 22 seconds  0.0.0.0:6033->3306/tcp, :::6033->3306/tcp      mariadb
```

Далее, через браузер заходим по адресу 10.0.1.12:8081 – видим приглашение phpmuadmin, вводим сервер – db, user - db_user, PASSWORD: db_user_pass, заканчиваем настройку.