Урок 7. Запуск веб-приложения из контейнеров

Установить в виртуальную машину или VDS Docker,

`sudo apt install docker.io` - установка docker;
`sudo apt install docker-compose` - установка docker-compose;
`sudo docker` - просмотр команд;
`sudo docker run hello-world` - создание контейнера и запуск;
`sudo docker ps -a` - просмотр всех контейнеров;
`sudo docker run --name my_hello hello-world` - создание контейнера с назначением имени;
`sudo docker images` - просмотр образов;
`sudo docker rm my_hello` - удаление контейнера;
`sudo docker rmi hello-world` - удаление образа;
`sudo docker search nginx` - поиск контейнера;
`sudo docker pull nginx` - установка официального контейнера nginx без запуска;
`sudo docker run -p 80:80 -d -v /var/www/html:/user/share/nginx/html --name nginx1 nginx` - запуск контейнера и проброс портов;
`sudo docker exec -ti nginx1 bash` - вход в контейнер;
`exit` - выход из контейнера после работы с ним;

настроить набор контейнеров через docker compose по инструкции по сылке: https://www.digitalocean.com/community/tutorials/how-to-install-wordpress-with-docker-compose-ru. Часть с настройкой certbot и HTTPS опустить, если у вас нет настоящего домена и белого IP.

Шаг 1 — Настройка конфигурации веб-сервера

```
mkdir wordress
cd wordress
```
- создаем директорию, переходим в нее;

`mkdir nginx-conf` - создайте директорию для файла конфигурации;
`nano nginx-conf/nginx.conf` - создаем конфигурационный файл, вставляем в него код:

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

        server_name example.com www.example.com;

        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;
        }
}
```

Шаг 2 — Настройка переменных среды

`nano .env` - откроем файл с именем .env;

```
MYSQL_ROOT_PASSWORD=1
MYSQL_USER=nadya
MYSQL_PASSWORD=1
```

- добавим в файл имена и значения переменных;

`nano .dockerignore` - создаем файл;

```
.env
docker-compose.yml
```

- прописываем данные, сохраняем;

Шаг 3 — Определение служб с помощью Docker Compose

`nano docker-compose.yml` - создаем файл, прописываем в него данные:

```
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
```

Шаг 4 — Получение сертификатов SSL и учетных данных

`sudo docker-compose up -d` - запуск контейнера в фоновом режиме;
`docker-compose ps` - проверка статуса служб;
127.0.0.1 –проверяем работу
`sudo docker stop webserver db wordpress` - останавливаем работу контейнеров;
`sudo docker rm webserver db wordpress` - удаление контейнеров
`sudo docker rmi nginx:1.15.12-alpine mysql:8.0 wordpress:5.1.1-fpm-alpine` - удаление образов;
`sudo docker ps -a` - просмотр всех контейнеров;
`sudo docker images` - просмотр образов;
`sudo docker volume ls` - просмотр созданных разделов;
`sudo docker volume rm wordress_dbdata wordress_wordpress` - удаляем ненужные разделы;
`sudo docker network ls` - просмотр сети;
`sudo docker network rm wordress_app-network` - удаление сети;

* 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).