

# Домашнее задание #9

---

## Критерии оценки

- 0 баллов - выполнено менее 60% пунктов задания.
- 2 балла - выполнено от 60% до 69% пунктов задания, отчет соответствует требованиям.
- 4 балла - выполнено от 70% до 89% пунктов задания, отчет соответствует требованиям.
- 6 баллов - выполнено более 90% пунктов задания.

## Оформление отчета

Отчет должен включать в себя файл `typescript.txt` полученный в результате выполнения команды:

```
$ script typescript.txt
```

Команду выше необходимо запустить перед выполнением домашнего задания. Для завершения записи достаточно выполнить команду `exit`. При выполнении команды `script` история команд Bash не сохраняется.

## Установка Docker

1. Запустите и подключитесь в виртуальной машине через SSH.
2. Далее необходимо произвести установку Docker Engine в соответствии с инструкцией:  
<https://docs.docker.com/engine/install/ubuntu/>
3. Для проверки корректности произведенной установки выполните следующую команду:

```
$ sudo docker run hello-world
```

## Образы (images)

1. Для того чтобы просмотреть список образов Docker, в системе, можно ввести следующую команду:

```
$ sudo docker image ls
```

2. Загрузка контейнеров осуществляется с помощью команды запуска:

```
$ sudo docker pull ubuntu
```

3. Чтобы удалить контейнер необходимо указать идентификатор(ID) образа:

```
$ sudo docker rmi <ID образа ubuntu>
```

4. Вывести только идентификаторы можно с помощью атрибута `-q`:

```
$ sudo docker images -q
```

5. Вывод истории образа контейнера осуществляется командой:

```
$ sudo docker history < ID образа>
```

## Запуск контейнера

1. Запуск контейнера ubuntu с подключением к оболочке ``bash`` в интерактивном режиме осуществляется следующей командой:

```
$ sudo docker run -it ubuntu /bin/bash
```

2. Запуск контейнера ubuntu в фоновом режиме осуществляется следующей командой:

```
$ sudo docker run -d -it ubuntu sleep 99999
```

3. Список запущенных контейнеров выводится следующим образом:

```
$ sudo docker ps
```

4. А для вывода всех контейнеров необходимо добавить к предыдущей команде атрибут `-a`:

```
$ sudo docker ps -a
```

5. А для вывода только идентификаторов всех контейнеров необходимо добавить к предыдущей команде атрибут `-q`:

```
$ sudo docker ps -qa
```

## Управление контейнерами

**Примечание:** `<ID контейнера>` - подразумевается контейнер запущенный в пункте 2 предыдущего раздела.

1. Для вывода списка процессов, запущенных в контейнере, необходимо ввести следующую команду:

```
$ sudo docker top <ID контейнера>
```

2. Остановка контейнера:

```
$ sudo docker stop <ID контейнера>
```

3. Старт контейнера:

```
$ sudo docker start <ID контейнера>
```

4. Вывести статистику использования ресурсов контейнера:

```
$ sudo docker stats <ID контейнера> --no-stream
```

5. Подключиться к контейнеру:

```
$ sudo docker exec -it <ID контейнера> /bin/bash
```

6. Для выхода из контейнера необходимо выполнить команду 'exit' или нажать сочетание `ctrl + d`.

7. Приостановить работу контейнера:

```
$ sudo docker pause <ID контейнера>
```

8. Возобновить работу контейнера:

```
$ sudo docker unpause <ID контейнера>
```

9. Удалить принудительно запущенный контейнер:

```
$ sudo docker rm -f <ID контейнера>
```

## Контейнеризация приложений

1. Установите утилиту Git на виртуальную машину и клонируйте репозиторий:

```
$ git clone https://github.com/docker/getting-started.git
```

2. Просмотрите содержимое клонированного репозитория. Внутри `getting-started/app` каталога вы должны увидеть `package.json` и два подкаталога (`src` и `spec`).
3. Создайте внутри директории `getting-started/app` файл с именем `Dockerfile` и запишите в него следующую информацию:

```
FROM node:18-alpine
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
```

4. Осуществите сборку пакета:

```
$ sudo docker build -t getting-started .
```

5. Запустите контейнер с приложением:

```
$ sudo docker run -dp 3000:3000 getting-started
```

6. Добавьте правило проброса для порта `3000` в настройках сети `NAT` виртуальной машины графического интерфейса VirtualBox.
7. Откройте страницу `http://localhost:3000` на хостовой машине. Должна открыться страница веб-приложения.
8. Остановите контейнер.

## Мультиконтейнерные приложения

1. Создайте сеть:

```
$ sudo docker network create todo-app
```

2. Запустите контейнер MySQL и подключите его к сети:

```
$ sudo docker run -d \
  --network todo-app --network-alias mysql \
  -v todo-mysql-data:/var/lib/mysql \
  -e MYSQL_ROOT_PASSWORD=secret \
  -e MYSQL_DATABASE=todos \
  mysql:8.0
```

3. Чтобы убедиться, что база данных функционирует штатно, подключитесь к базе:

```
$ sudo docker exec -it <ID контейнера MYSQL> mysql -u root -p
```

4. Далее введите пароль `secret` и убедитесь, что база `todos` существует:

```
mysql> SHOW DATABASES;
```

5. Выйдите из оболочки SQL:

```
mysql> exit
```

6. Далее запустите новый контейнер, используя образ `nicolaka/netshoot`. Обязательно подключите его к той же сети:

```
$ sudo docker run -it --network todo-app nicolaka/netshoot
```

7. Проверьте доступность сервиса `mysql` запустив команду `dig` внутри контейнера:

```
$ dig mysql
```

8. Завершите все запущенные контейнеры.

## Docker compose

1. Проверьте установлен ли `docker compose`:

```
$ sudo docker compose version
```

2. Создайте в директории `~/getting-started/app` файл с именем `docker-compose.yml`:

3. В первую очередь необходимо определиться с именем сервиса и образом контейнера, сделав соответствующую запись в файле:

```
services:
  app:
    image: node:18-alpine
```

4. Далее необходимо добавить команду, которая будет выполнена после запуска контейнера

```
services:
  app:
    image: node:18-alpine
    command: sh -c "yarn install && yarn run dev"
```

5. Также необходимо указать правила для порта 3000:

```
services:
  app:
    image: node:18-alpine
    command: sh -c "yarn install && yarn run dev"
    ports:
      - 3000:3000
```

6. После необходимо указать рабочую директорию `working_dir` и том `volume`, в котором будет храниться код веб-приложения:

```
services:
  app:
    image: node:18-alpine
    command: sh -c "yarn install && yarn run dev"
    ports:
      - 3000:3000
    working_dir: /app
    volumes:
      - ./:/app
```

7. Наконец, необходимо указать переменные окружения `enviroment` для подключения к базе данных:

```
services:
  app:
    image: node:18-alpine
    command: sh -c "yarn install && yarn run dev"
```

```
ports:
  - 3000:3000
working_dir: /app
volumes:
  - ./:/app
environment:
  MYSQL_HOST: mysql
  MYSQL_USER: root
  MYSQL_PASSWORD: secret
  MYSQL_DB: todos
```

8. Для подключения к базе данных необходимо указать еще один сервис и соответствующий образ в файле:

```
services:
  app:
    # параметры сервиса app
  mysql:
    image: mysql:8.0
```

9. Далее необходимо определить том для сервиса `mysql`:

```
services:
  app:
    # параметры сервиса app
  mysql:
    image: mysql:8.0
    volumes:
      - todo-mysql-data:/var/lib/mysql

volumes:
  todo-mysql-data:
```

10. После указать переменные окружения:

```
services:
  app:
    # параметры сервиса app
  mysql:
    image: mysql:8.0
    volumes:
      - todo-mysql-data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: todos
```

```
volumes:
  todo-mysql-data:
```

11. Итоговый файл `docker-compose.yml` должен выглядеть следующим образом:

```
services:
  app:
    image: node:18-alpine
    command: sh -c "yarn install && yarn run dev"
    ports:
      - 3000:3000
    working_dir: /app
    volumes:
      - ./:/app
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: root
      MYSQL_PASSWORD: secret
      MYSQL_DB: todos

  mysql:
    image: mysql:8.0
    volumes:
      - todo-mysql-data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: todos

volumes:
  todo-mysql-data:
```

12. Выведите содержимое файла `docker-compose.yml`:

```
$ cat docker-compose.yml
```

13. Для запуска приложения необходимо выполнить следующую команду:

```
$ sudo docker compose up -d
```

14. Откройте страницу `http://localhost:3000` на хостовой машине. Должна открыться страница веб-приложения.

15. Остановите и удалите запущенные сервисы, а также связанные с ними тома:

```
$ sudo docker compose down -v
```



