Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Новосибирский государственный технический университет»

Кафедра теоретической и прикладной информатики

Лабораторная работа №2
по дисциплине «Разработка web-приложений и распределенных информационных систем»

Факультет: ПМИ

Группа: ПМ-92

Студент: Иванов В. В.

Преподаватель: Цыгулин А. А.

Новосибирск

2022

Задание

Создать комплекс контейнеров (Docker Compose) и разместить их в репозитории Docker Hub. При загрузке комплекса должен запуститься сервер, предоставляющий доступ к странице сайта.

Ход работы

Для выполнения работы создадим сервер на Python, позволяющий отображать содержимое текстового файла на сайте, а также клиентское приложение, создающее запросы.

Создаём рабочую директорию:

```
~/Docker/lab2 ls
client  docker-compose.yml  server
```

1. **server**

```
~/Docker/lab2/server ls
Dockerfile  index.html  server.py
```

Создаём сервер (**server.py**):

```
#!/usr/bin/env python3
import http.server
import socketserver
handler = http.server.SimpleHTTPRequestHandler
with socketserver.TCPServer(("", 1234), handler) as httpd:
  httpd.serve_forever()
```

В **index.html** добавим строку:

```
Hello World
```

Dockerfile создадим на основе образа для Python-приложений:

```
FROM python:latest
ADD server.py /server/
ADD index.html /server/
WORKDIR /server/
```

2. client

Создаём клиент (**client.py**):

```python
import urllib.request
fp = urllib.request.urlopen("http://localhost:1234/")
encodedContent = fp.read()
decodedContent = encodedContent.decode("utf8")
print(decodedContent)
fp.close()
```

Создаём **Dockerfile** для клиента:

```
FROM python:latest
ADD client.py /client/
WORKDIR /client/
```

3. docker-compose.yml

Теперь объединим контейнеры, добавив в **docker-compose.yml** (ссылка на файл) следующее содержимое:

```yaml
version: "3"
services:
  server:
    build: server/
    command: python ./server.py
    ports:
      - 1234:1234
  client:
    build: client/
    command: python ./client.py
    network_mode: host
    depends_on:
      - server
```

Здесь мы имеем два сервиса: **server** и **client**. В обоих сервисах путь к **Dockerfile** указывается при помощи команды **build**. Поле **command** указывает на запуск соответствующих компонентов. Также задается порт, ожидание запуска сервера клиентом (**depends_on**) и возможность обращения к localhost (**network_mode**).

4. Создание образа и тестирование

В рабочей директории выполняем сборку:

```
~/Docker/lab2 sudo docker-compose build
[+] Building 10.2s (8/14)
 => [lab2_server internal] load build definition from Dockerfile
 => => transferring dockerfile: 123B
 => [lab2_client internal] load build definition from Dockerfile
 => => transferring dockerfile: 98B
 => [lab2_client internal] load .dockerignore
 => => transferring context: 2B
 => [lab2_server internal] load .dockerignore
 => => transferring context: 2B
```

Запускаем комплекс:

```
~/Docker/lab2 sudo docker-compose up
[+] Running 3/2
 ⠿ Network lab2_default      Created
 ⠿ Container lab2-server-1   Created
 ⠿ Container lab2-client-1   Created
Attaching to lab2-client-1, lab2-server-1
lab2-server-1  | 172.18.0.1 - - [23/May/2022 19:34:52] "GET / HTTP/1.1" 200
lab2-client-1  | Hello World
lab2-client-1  |
lab2-client-1  |
lab2-client-1  |
lab2-client-1 exited with code 0
```

Смотрим запущенные сервисы:

```
~/Docker/lab2 sudo docker-compose ps
NAME                COMMAND                  SERVICE             STATUS              PORTS
lab2-client-1       "python ./client.py"     client              exited (0)
lab2-server-1       "python ./server.py"     server              running             0.0.0.0:1234
->1234/tcp, :::1234->1234/tcp
~/Docker/lab2
```

Переходим на localhost и видим результат:

Hello World

Загружаем контейнеры на Docker Hub:

sudo docker build -t lenferdetroud/server server
sudo docker build -t lenferdetroud/client client
sudo docker push lenferdetroud/server:latest
sudo docker push lenferdetroud/client:latest