МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения
(ИиППО)

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 9

«Межконтейнерное взаимодействие docker-контейнеров через localhost»
по дисциплине
«Настройка и администрирование сервисного программного обеспечения»

Выполнил студент группы ИКБО-30-21 Смирнов А.Е.

Принял преподаватель Благирев М.М

Практическая работа выполнена «__»_____2023 г. *(подпись студента)*

«Зачтено» «__»_____2023 г. *(подпись руководителя)*

Москва 2023

Цель работы: получить навыки создания сетевого соединения и передачи данных между docker-контейнерами посредством localhost.

Теоретическое введение.

Docker создает сеть по умолчанию, в которой запущены все контейнеры, и задает имя сети для каждого контейнера, используя имя контейнера.

Для межконтейнерного взаимодействия можно использовать брокер сообщений Apache ActiveMQ. Если есть mq для ActiveMQ, то можно использовать так: tcp://mq:61616 (или любой другой настроенный протокол/порт) из других контейнеров, чтобы подключиться к нему.

При этом не нужно устанавливать параметр --net если не нужно создавать определенную сеть для использования определенных контейнеров.

По умолчанию в Docker включено межконтейнерное взаимодействие, это означает, что все контейнеры могут взаимодействовать между собой (используя сеть docker0). Эта возможность может быть отключена путём запуска Docker сервиса с флагом --icc=false.

Для сетевого взаимодействия контейнеров можно сделать следующее:

1) создать новую сеть

$ docker network create <network-name>

2) Подключить контейнеры к сети

$ docker run --net=<network-name> ...

или

$ docker network connect <network-name> <container-name>

3) пинг контейнер по имени

docker exec -ti <container-name-A> ping <container-name-B>

Задание на практическую работу

Создать два докер-контейнера, которые будут одновременно "слушать" localhost на разных портах. Пользователь делает запрос к контейнеру №1. Далее контейнер №1 делает запрос к контейнеру №2. Контейнер №2 обработает запрос, передаст результат контейнеру №1. Контейнер №1 передаст результат пользователю. API может быть реализовано на любой технологии. В итоге необходимо настроить систему из нескольких контейнеров, связанных друг с другом.

Листинг 1 - Файл docker-compose.yml

```
version: '3'
services:
  container1:
    build: ./container1
    ports:
      - "8000:8000"

  container2:
    build: ./container2
    ports:
      - "5000:5000"
```

Листинг 2 - Файл app.py директории container1

```
from flask import Flask
import requests

app = Flask(__name__)

@app.route('/')
def index():
    response = requests.get('http://container2:5000')
    return response.text

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8000)
```

Листинг 3 - Файл Dockerfile директории container1

```
FROM python:3.9
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
EXPOSE 8000
CMD ["python", "app.py"]
```

Листинг 4 - Файл requirements.txt директории container1

```
flask==2.0.1
requests==2.26.0
```

Листинг 5 - Файл app.js директории container2

```
const express = require('express');
const app = express();

app.get('/', (req,res) => {
   res.end('Hello from Container 2!')
});

app.listen(5000, () => {
   console.log('Server running on port 5000');
});
```

Листинг 6- Файл Dockerfile директории container2

```
FROM node:14
WORKDIR /app
RUN npm install
COPY . .
EXPOSE 5000
CMD ["node", "app.js"]
```

Листинг 7- Файл package.json директории container2

```
{
   "dependencies": {
      "express": "^4.17.1",
      "axios": "^0.21.1"
   }
```

```
}
```

```
[+] Building 7.6s (20/20) FINISHED
 => [niaspo9pract-container1 internal] load build definition from Dockerfile                                    0.0s
 => => transferring dockerfile: 32B                                                                             0.0s
 => [niaspo9pract-container2 internal] load build definition from Dockerfile                                    0.0s
 => => transferring dockerfile: 32B                                                                             0.0s
 => [niaspo9pract-container1 internal] load .dockerignore                                                       0.0s
 => => transferring context: 2B                                                                                 0.0s
 => [niaspo9pract-container2 internal] load .dockerignore                                                       0.0s
 => => transferring context: 2B                                                                                 0.0s
 => [niaspo9pract-container1 internal] load metadata for docker.io/library/python:3.9                          0.6s
 => [niaspo9pract-container2 internal] load metadata for docker.io/library/node:14                              0.5s
 => [niaspo9pract-container2 internal] load build context                                                       0.0s
 => => transferring context: 127B                                                                               0.0s
 => [niaspo9pract-container2 1/6] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4  0.0s
 => CACHED [niaspo9pract-container2 2/6] WORKDIR /app                                                           0.0s
 => [niaspo9pract-container1 internal] load build context                                                       0.0s
 => => transferring context: 93B                                                                                0.0s
 => [niaspo9pract-container1 1/5] FROM docker.io/library/python:3.9@sha256:6ea9dafc96d7914c5c1d199f1f0195c4e05cf0  0.0s
 => [niaspo9pract-container2 3/6] COPY package.json .                                                           0.4s
 => CACHED [niaspo9pract-container1 2/5] WORKDIR /app                                                           0.0s
 => [niaspo9pract-container1 3/5] COPY requirements.txt .                                                       0.4s
 => [niaspo9pract-container2 4/6] COPY package-lock.json .                                                      0.1s
 => [niaspo9pract-container1 4/5] RUN pip install --no-cache-dir -r requirements.txt                            6.2s
 => [niaspo9pract-container2 5/6] RUN npm install                                                               4.9s
 => [niaspo9pract-container2 6/6] COPY . .                                                                      0.0s
 => [niaspo9pract-container1] exporting to image                                                                0.3s
 => => exporting layers                                                                                         0.2s
 => => writing image sha256:575f04bd32059ad68922d321bd15f7eebb8f31ac94c274f4fb18e4da15a9be2b                    0.0s
 => => naming to docker.io/library/niaspo9pract-container2                                                      0.0s
 => => writing image sha256:ce61a56f2dc0b4c5e1e28dd0f617dc3dffe1a7edc8ef2c8225922b4267bc6e7e                    0.0s
 => => naming to docker.io/library/niaspo9pract-container1                                                      0.0s
 => [niaspo9pract-container1 5/5] COPY . .                                                                      0.0s
[+] Running 3/3
 - Network niaspo9pract_default         Created                                                                 0.1s
 - Container niaspo9pract-container2-1  Created                                                                 0.1s
 - Container niaspo9pract-container1-1  Created                                                                 0.1s
Attaching to niaspo9pract-container1-1, niaspo9pract-container2-1
niaspo9pract-container2-1  | Server running on port 5000
niaspo9pract-container1-1  |  * Serving Flask app 'app' (lazy loading)
niaspo9pract-container1-1  |  * Environment: production
niaspo9pract-container1-1  |    WARNING: This is a development server. Do not use it in a production deployment.
niaspo9pract-container1-1  |    Use a production WSGI server instead.
niaspo9pract-container1-1  |  * Debug mode: off
niaspo9pract-container1-1  | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
niaspo9pract-container1-1  |  * Running on all addresses (0.0.0.0)
niaspo9pract-container1-1  |  * Running on http://127.0.0.1:8000
niaspo9pract-container1-1  |  * Running on http://172.20.0.3:8000
niaspo9pract-container1-1  | Press CTRL+C to quit
niaspo9pract-container1-1  | 172.20.0.1 - - [11/May/2023 21:53:00] "GET / HTTP/1.1" 200 -
niaspo9pract-container1-1  | 172.20.0.1 - - [11/May/2023 21:53:01] "GET /favicon.ico HTTP/1.1" 404 -
niaspo9pract-container1-1  | 172.20.0.1 - - [11/May/2023 21:53:44] "GET / HTTP/1.1" 200 -
```

Рисунок 1 – Результат выполнения команды docker-compose up

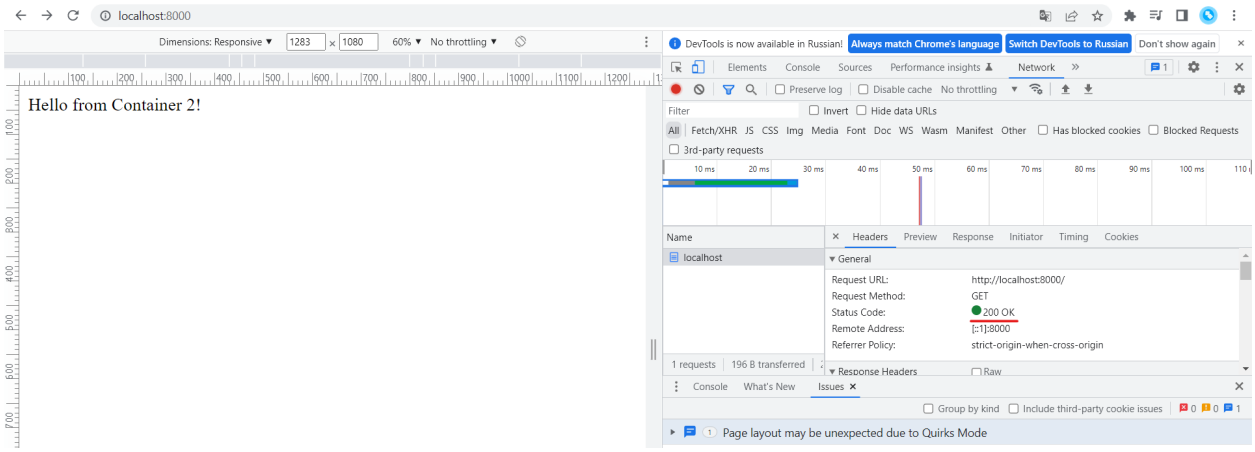

Рисунок 2 – Подтверждение выполненного GET-запроса

Вывод:

В ходе выполнения данной практической работы были получены навыки создания сетевого соединения и передачи данных между docker-контейнерами посредством locallhost.

Список литературы

1. Соединение контейнеров опцией --link. – URL: https://docs.docker.com/network/links/ (дата обращения: 03.05.2023).

2. Обзор сетевых возможностей Docker. – URL: https://docs.docker.com/network/ (дата обращения: 03.05.2023).

3. Сейерс, Э. Х. Docker на практике / Э. Х. Сейерс, А. Милл ; перевод с английского Д. А. Беликов. — Москва: ДМК Пресс, 2020. — 516 с. — ISBN 978-5-97060-772-5. — Текст : электронный // Лань : электроннобиблиотечная система. — URL: https://e.lanbook.com/book/131719 (дата обращения: 03.05.2023). — Режим доступа: для авториз. пользователей.