

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 4

Выполнил: Рашевский Вячеслав
Васильевич
Группа К33402

Проверил:
Добряков Д. И.

Санкт-Петербург

Задача:

Необходимо упаковать ваше приложение в docker-контейнеры и обеспечить сетевое взаимодействие между различными частями вашего приложения, а также настроить общение микросервисов между собой посредством RabbitMQ. Делать это можно как с помощью dockercompose так и с помощью docker swarm. При разумном использовании swirl вы получите дополнительные баллы.

1. Построим прокси на nginx, свяжем сервисы приложения и разделим функционал:

```
7 http {
8     include      mime.types;
9     default_type  application/octet-stream;
10    sendfile      on;
11
12    upstream app {
13        server app:3000;
14    }
15
16    upstream product_service {
17        server product-service:3001;
18    }
19
20    server {
21        listen 80;
22
23        location / {
24            proxy_pass http://app;
25            proxy_set_header Host $host;
26            proxy_set_header X-Real-IP $remote_addr;
27            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
28            proxy_set_header X-Forwarded-Proto $scheme;
29        }
30
31        location /products {
32            proxy_pass http://product_service;
33            proxy_set_header Host $host;
34            proxy_set_header X-Real-IP $remote_addr;
35            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
36            proxy_set_header X-Forwarded-Proto $scheme;
37        }
38    }
39 }
40
```

В каждом сервисе создадим докерфайл, и распишем docker-compose.yml для сборки сервисов, подключения postgresql и nginx:

```
1 version: '3.9'
2
3 services:
4   lr2:
5     container_name: lr2
6     build:
7       context: ./lr2
8       dockerfile: Dockerfile
9     ports:
10      - "3000"
11     depends_on:
12      - postgres
13     environment:
14      - DATABASE_URL=postgres://postgres:postgres@postgres:5432/db
15     volumes:
16      - ./lr2:/usr/src/app:cached
17      - /usr/src/app/node_modules
18     networks:
19      - my-network
20
21   lr3:
22     container_name: lr3
23     build:
24       context: ./lr3
25       dockerfile: Dockerfile
26     ports:
27      - "3001"
28     depends_on:
29      - postgres
30     environment:
31      - DATABASE_URL=postgres://postgres:postgres@postgres:5432/db
32     volumes:
33      - ./lr3:/usr/src/app:cached
34      - /usr/src/app/node_modules
35     networks:
36      - my-network
37
38   nginx:
39     container_name: nginx
40     image: nginx:alpine
41     ports:
42      - "80:80"
43     depends_on:
44      - lr2
45      - lr3
46     volumes:
47      - ./nginx.conf:/etc/nginx/nginx.conf
48     networks:
49      - my-network
50
51   postgres:
52     container_name: postgres
53     image: postgres
54     environment:
55      POSTGRES_USER: postgres
56      POSTGRES_PASSWORD: postgres
57      POSTGRES_DB: db
```

```

58     volumes:
59       - pgdata:/var/lib/postgresql/data
60     networks:
61       - my-network
62
63 networks:
64   my-network:
65     driver: bridge
66
67 volumes:
68   pgdata:
69

```

4. Для каждой папки с докер файлом распишем докер-игнор:

```

1  node_modules
2  app/node_modules
3  product-service/node_modules
4  dist
5

```

5. Запуск и проверка работы

<input type="checkbox"/>		lr4		Exited		N/A	5 hours ago		:		
<input type="checkbox"/>		lr3-1	67c7085c118d	lr4-lr3	Exited (137)	0:3001	N/A	5 hours ago		:	
<input type="checkbox"/>		lr2-1	c8147e21b88b	lr4-lr2	Exited (137)	0:3000	N/A	5 hours ago		:	
<input type="checkbox"/>		nginx-1	873b00b4f9b1	nginx.alpine	Exited	80:80	N/A	5 hours ago		:	
<input type="checkbox"/>		postgres-1	259d86ec374b	postgres.alpine	Exited (137)		N/A	5 hours ago		:	

Регистрация

docker check / reg

Save
 Share

POST

http://localhost/register

Send

Params

Authorization

Headers (8)

Body

Scripts

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

```

1  {
2    "email": "test@example.com",
3    "password": "password123",
4    "isAdmin": true
5  }

```

Body

Cookies

Headers (7)

Test Results

201 Created 82 ms 443 B

Save as example

Pretty

Raw

Preview

Visualize

JSON

```

1  {
2    "id": 14,
3    "email": "test@example.com",
4    "password": "$2b$10$ayz/2WVf11x0g2EjCN0jz0d5.NTNeDOUm1EtX3mNQDCxivKW7myZ0",
5    "isAdmin": true,
6    "updatedAt": "2024-06-13T20:35:57.315Z",
7    "createdAt": "2024-06-13T20:35:57.315Z"
8  }

```

Вход

HTTP

docker check / log

Save

Share

POST

http://localhost/login

Send

Params

Authorization

Headers (8)

Body

Scripts

Settings

Cookies

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

Beautify

```
1 {
2   "email": "test@example.com",
3   "password": "password123"
4 }
```

Body

Cookies

Headers (7)

Test Results

200 OK

63 ms

406 B

Save as example

...

Pretty

Raw

Preview

Visualize

JSON

Copy

Find

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTesImIzQWRtaW4iOnRydWUsImIhdCI6MTcxODMxMDk3OSwiZSwiOiJjoxNzE4MzE0NTc5fQ.6TyYkssUEwFvLFb4UXQYkz1uIEpm5jC6qVR4RSq0hM8"
3 }
```

Добавление товара:

HTTP

docker check / post prod

Save

Share

POST

http://localhost/products

Send

Params

Authorization

Headers (9)

Body

Scripts

Settings

Cookies

Auth Type

Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

Token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...

Body

Cookies

Headers (7)

Test Results

201 Created 22 ms 441 B

Save as example

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "discount": 0,
3   "id": 3,
4   "name": "Клавиатура 80 клавиш",
5   "category": "Периферия",
6   "price": 3500,
7   "stock": 15,
8   "updatedAt": "2024-06-13T20:36:56.991Z",
9   "createdAt": "2024-06-13T20:36:56.991Z"
10 }
```

Поиск по id

HTTP

docker check / get

Save

Share

PUT

http://localhost/products/2

Send

Params

Authorization

Headers (8)

Body

Scripts

Settings

Cookies

Auth Type

Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

Token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...

Body

Cookies

Headers (7)

Test Results

200 OK

18 ms

436 B

Save as example

...

Pretty

Raw

Preview

Visualize

JSON

...

...

```
1  {
2    "id": 2,
3    "name": "Клавиатура 80 клавиш",
4    "category": "Периферия",
5    "price": 3500,
6    "stock": 15,
7    "discount": 0,
8    "createdAt": "2024-06-10T18:38:16.316Z",
9    "updatedAt": "2024-06-10T18:38:16.316Z"
10 }
```

Поиск

HTTP docker check / search Save Share

GET ▼ http://localhost/products/search?query=Периферия Send ▼

Params ● Authorization Headers (6) Body Scripts Settings Cookies

Query Params

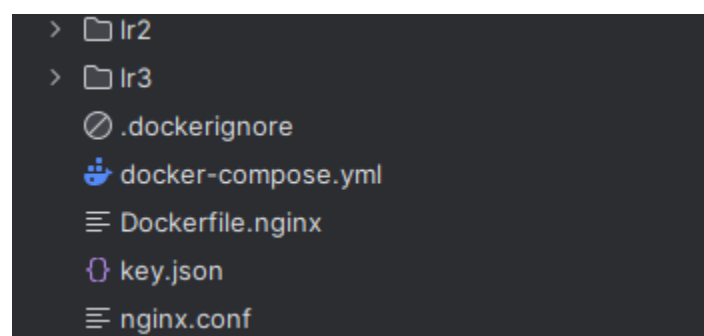
<input checked="" type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	query	Периферия			
	Key	Value	Description		

Body Cookies Headers (7) Test Results 200 OK 5 ms 843 B Save as example ...

Pretty Raw Preview Visualize JSON ▼ 🔍

```
1 [
2   {
3     "id": 1,
4     "name": "Клавиатура 80 клавиш",
5     "category": "Периферия",
6     "price": 3500,
7     "stock": 15,
8     "discount": 0,
9     "createdAt": "2024-06-10T17:58:35.088Z",
10    "updatedAt": "2024-06-10T17:58:35.088Z",
11  },
12  {
13    "id": 2,
14    "name": "Клавиатура 80 клавиш",
15    "category": "Периферия",
16    "price": 3500,
17    "stock": 15,
18    "discount": 0,
19    "createdAt": "2024-06-10T18:38:16.316Z",
20    "updatedAt": "2024-06-10T18:38:16.316Z",
21  }
22 ]
```

Структура проекта:



Вывод:

В рамках лабораторной работы №4 было успешно упаковано приложение в Docker-контейнеры и настроено сетевое взаимодействие между его компонентами. Также было обеспечено общение микросервисов через Nginx, используя Docker Compose для оркестрации, а Docker Swarm использован для повышения масштабируемости и надежности системы.