

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

**Лабораторная работа №4
(Реализация через Docker Swarm)**

Выполнил:

Таначев Егор

Группа К33412

Проверил:

Добряков Д. И.

Санкт-Петербург

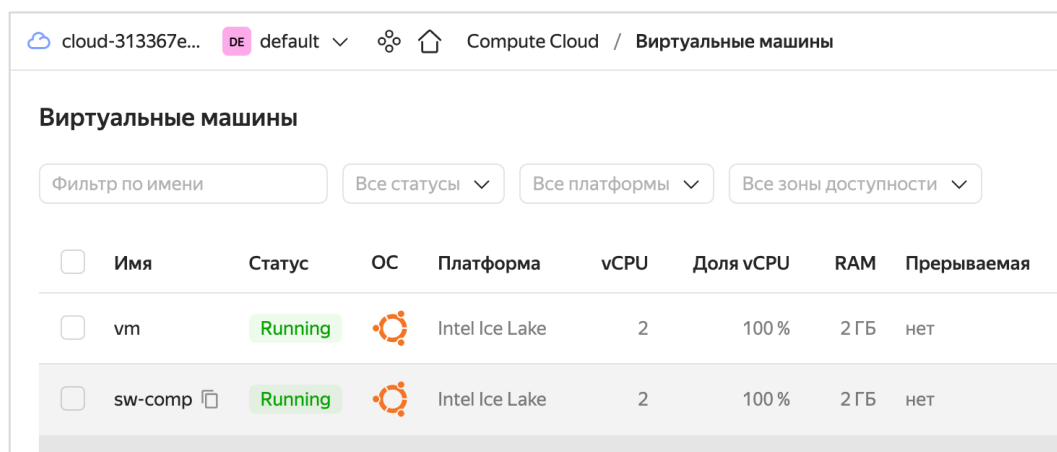
2023 г.

Задача

Упаковать наше приложение в docker-контейнеры и обеспечить сетевое взаимодействие между различными частями вашего приложения. Делать это можно как с помощью docker-compose, так и с помощью docker swarm.

Ход работы

Создадим виртуальную машину с Ubuntu 22.04 на Yandex.Cloud, как показано на Рисунке 1. Зададим название sw-comp и сгенерируем для нее ssh ключ с помощью команды `ssh-keygen -t ed25519`.



| Виртуальные машины | | | | | | | |
|----------------------------------|---------|--------------|----------------|---------------|-----------|----------------------|-------------|
| Фильтр по имени | | Все статусы | | Все платформы | | Все зоны доступности | |
| <input type="checkbox"/> Имя | Статус | ОС | Платформа | vCPU | Доля vCPU | RAM | Прерываемая |
| <input type="checkbox"/> vm | Running | Ubuntu 22.04 | Intel Ice Lake | 2 | 100 % | 2 ГБ | нет |
| <input type="checkbox"/> sw-comp | Running | Ubuntu 22.04 | Intel Ice Lake | 2 | 100 % | 2 ГБ | нет |

Рисунок 1 – Виртуальная машина sw-comp на Yandex.Cloud

Теперь локально подготовим наше приложение для того, чтобы потом его просто клонировать и сразу запустить с помощью docker swarm.

Для этого создадим файл docker-swarm. yaml, как показано на Рисунке 2, который будет определять нашу инфраструктуру, состоящую из сервисов: User и Market. В этом файле, в отличие от реализации через Docker Compose, мы указываем имя образа, порт, команды для запуска, сеть и количество реплик (для стабилизации нагрузки).

```

1  version: "3.9"
2
3  services:
4    user:
5      image: user-service
6      ports:
7        - "1111:1111"
8      command: npm run start
9      networks:
10       - app-network
11      deploy:
12        replicas: 3
13
14    market:
15      image: market-service
16      ports:
17        - "2222:2222"
18      command: npm run start
19      networks:
20       - app-network
21      deploy:
22        replicas: 3
23
24  networks:
25    app-network:
26      driver: overlay

```

Рисунок 2 – Файл docker-swarm.yaml

Для каждой части приложения создадим Dockerfile, который определяет, как упаковать эту часть в контейнер. Dockerfile содержит инструкции по установке зависимостей, копированию файлов и настройке окружения. Dockerfile показан на Рисунках 3-4.

```
user > Dockerfile
1  # На каком образе работает докер контейнер
2  FROM node:16-alpine
3
4  # Рабочая директория
5  WORKDIR /app
6
7  # Копирование package.json и package-lock.json
8  COPY package*.json ./
9
10 # Установка зависимостей
11 RUN npm install
12
13 # Копирование всех файлов
14 COPY . .
15
16 # Прослушивание порта
17 EXPOSE 1111
18
19 # Запуск скрипта
20 CMD [ "npm", "start" ]
```

Рисунок 3 – Dockerfile в User

```
market > Dockerfile
1  # На каком образе работает докер контейнер
2  FROM node:16-alpine
3
4  # Рабочая директория
5  WORKDIR /app
6
7  # Копирование package.json и package-lock.json
8  COPY package*.json ./
9
10 # Установка зависимостей
11 RUN npm install
12
13 # Копирование всех файлов
14 COPY . .
15
16 # Прослушивание порта
17 EXPOSE 2222
18
19 # Запуск скрипта
20 CMD [ "npm", "start" ]
```

Рисунок 4 – Dockerfile в Market

Теперь подключимся к нашей виртуальной машине с помощью ssh, установим docker и склонируем наше приложение с GitHub. После этого

необходимо сбилдить образы командой *docker build -t <имя_образа>* для каждого микросервиса, как показано на Рисунке 5-6.

```
admin@sw-comp:~/ITMO-ICT-Backend-2023/labs/K33412/Tanachev Egor/LW4/Docker Swarm/task/user$ docker build -t user-service .
[+] Building 2.9s (10/10) FINISHED
=> [internal] load .dockerignore                                0.3s
=> => transferring context: 2B                                  0.0s
=> [internal] load build definition from Dockerfile             0.3s
=> => transferring dockerfile: 497B                             0.0s
=> [internal] load metadata for docker.io/library/node:16-alpine 1.7s
=> [1/5] FROM docker.io/library/node:16-alpine@sha256:6c381d5dc2a11dcdb693f0301e8587e43f440c90cdb8933eaaabb905d44cdb9 0.0s
=> [internal] load build context                                0.1s
=> => transferring context: 235.46kB                             0.0s
=> CACHED [2/5] WORKDIR /app                                    0.0s
=> CACHED [3/5] COPY package*.json ./                          0.0s
=> CACHED [4/5] RUN npm install                                 0.0s
=> [5/5] COPY . .                                              0.3s
=> exporting to image                                           0.2s
=> => exporting layers                                           0.1s
=> => writing image sha256:be255a18db2ae6366e9cf6e80c535c92ac4ccfbbbc8d53998457731b78d8cabe 0.0s
=> => naming to docker.io/library/user-service                  0.0s
```

Рисунок 5 – Создание образа для User-service

```
admin@sw-comp ~/ITMO-ICT-Backend-2023/labs/K33412/Tanachev Egor/LW4/Docker Swarm/task/market$ docker build -t market-service .
[+] Building 1.2s (10/10) FINISHED
=> [internal] load build definition from Dockerfile             0.1s
=> => transferring dockerfile: 497B                             0.0s
=> [internal] load .dockerignore                                0.1s
=> => transferring context: 2B                                  0.0s
=> [internal] load metadata for docker.io/library/node:16-alpine 0.5s
=> [1/5] FROM docker.io/library/node:16-alpine@sha256:6c381d5dc2a11dcdb693f0301e8587e43f440c90cdb8933eaaabb905d44cdb9 0.0s
=> [internal] load build context                                0.1s
=> => transferring context: 256.43kB                             0.1s
=> CACHED [2/5] WORKDIR /app                                    0.0s
=> CACHED [3/5] COPY package*.json ./                          0.0s
=> CACHED [4/5] RUN npm install                                 0.0s
=> [5/5] COPY . .                                              0.2s
=> exporting to image                                           0.1s
=> => exporting layers                                           0.1s
=> => writing image sha256:9e8b1b7a59baa0abab8187e2d2753bb67f52c962e1f754ae3180ba8975567b87 0.0s
=> => naming to docker.io/library/market-service                0.0s
```

Рисунок 6 – Создание образа для Market-service

Теперь развернем наше приложение с помощью команды *docker stack deploy -c docker-swarm.yaml api-market*, как показано на Рисунке 7.

```
admin@sw-comp ~/ITMO-ICT-Backend-2023/labs/K33412/Tanachev Egor/LW4/Docker Swarm/task$ docker stack deploy -c docker-swarm.yaml api-market
Creating network api-market_app-network
Creating service api-market_user
Creating service api-market_market
```

Рисунок 7 – Команда docker-compose build

И мы даже можем его проверить, если перейдем по публичному IP-адресу нашей виртуальной машины, как показано на Рисунке 8.

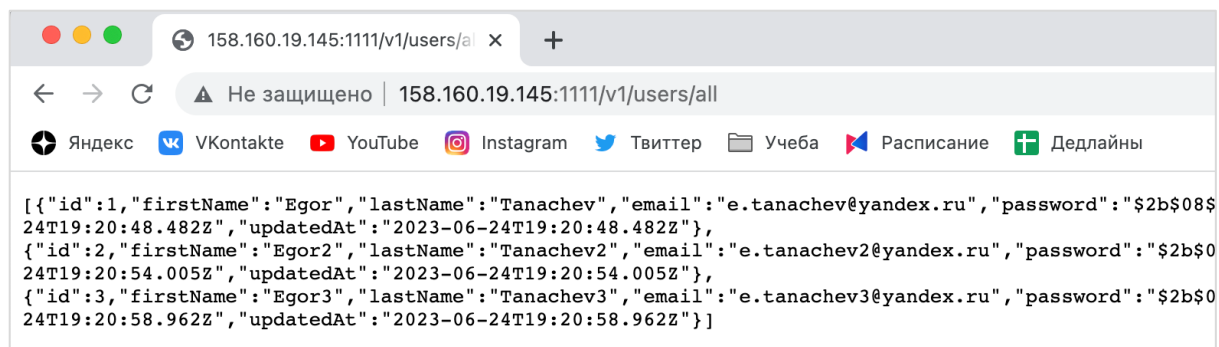


Рисунок 8 – Команда docker-compose up

Вывод

В результате работы была выполнена упаковка приложения на основе Express.js в Docker-контейнеры и обеспечено сетевое взаимодействие между двумя микросервисами: "user" и "market". Для достижения этой цели был использован Docker Swarm.