САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бек-энд разработка

Отчет

Лабораторная работа 4

Выполнил:

Дао Куанг Ань

Группа К33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2022 г.

Задача:

- Упаковать ваше 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

user/Dockerfile

```
FROM node


WORKDIR /app/attendance


COPY package.json .


RUN npm install


COPY . .


EXPOSE 8003


CMD ["npm", "start"]
```

event/Dockerfile

```
FROM node


WORKDIR /app/event
```

```
COPY package.json .


RUN npm install


COPY . .


EXPOSE 8002


CMD ["npm", "start"]
```

user/Dockerfile

```
FROM node


WORKDIR /app/user


COPY package.json .


RUN npm install


COPY . .


EXPOSE 8001


CMD ["npm", "start"]
```

docker-compose.yml

```yaml
version: '3'

services:


  nosql-db:

    image: mongo

    ports:

      - "27017:27017"

    container_name: nosql-db

    volumes:

      - ./db/:/data/db


  event:

    build:

      dockerfile: Dockerfile

      context: ./event

    container_name: event

    ports:

      - "8002:8002"

    restart: always

    depends_on:

      - "nosql-db"

    volumes:

      - .:/app

      - /app/event/node_modules
```

```yaml
user:
  build:
    dockerfile: Dockerfile
    context: ./user
  container_name: user
  ports:
    - "8001:8001"
  restart: always
  depends_on:
    - "nosql-db"
  volumes:
    - .:/app
    - /app/user/node_modules


attendance:
  build:
    dockerfile: Dockerfile
    context: ./attendance
  container_name: attendance
  ports:
    - "8003:8003"
  restart: always
  depends_on:
    - "nosql-db"
  volumes:
    - .:/app
    - /app/attendance/node_modules
```

```
  nginx-proxy:
    build:
      dockerfile: Dockerfile
      context: ./proxy
    depends_on:
      - event
      - attendance
      - user
    ports:
      - "80:80"
```

proxy/Dockerfile

```
FROM nginx

RUN rm /etc/nginx/nginx.conf

COPY nginx.conf /etc/nginx/nginx.conf
```

proxy/nginx.conf

```
worker_processes 4;

events { worker_connections 1024; }

http {
```

```
server {

    listen 80;
    charset utf-8;

    location / {
        proxy_pass http://event:8002;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'Upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    location ~ ^/attendance {
        rewrite ^/attendance/(.*) /$1 break;
        proxy_pass http://attendance:8003;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'Upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
```

```
        location ~ ^/user {

            rewrite ^/user/(.*) /$1 break;

            proxy_pass http://user:8001;

            proxy_http_version 1.1;

            proxy_set_header Upgrade $http_upgrade;

            proxy_set_header Connection 'Upgrade';

            proxy_set_header Host $host;

            proxy_cache_bypass $http_upgrade;

        }

    }

}
```

Вывод

- Упаковано приложение в docker-контейнеры и обеспечено сетевое взаимодействие между различными частями вашего приложения. Делать это можно как с помощью docker-compose так и с помощью docker swarm.