САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бек-энд разработка

Отчет
Лабораторная работа №4

Выполнил: Митурский Богдан Антонович

Группа: K33392

Проверил:
Добряков Д. И.

Санкт-Петербург

2024 г.

Задача
Необходимо упаковать ваше приложение в docker-контейнеры и обеспечить сетевое взаимодействие между различными частями вашего приложения, а также настроить общение микросервисов между собой.

Ход работы
Подготовим docker compose для уже созданной структуры:

```
name: sheep-royale-2
services:
 backend:
   container_name: "${COMPOSE_PROJECT_NAME}_backend"
   depends_on:
     - mongo
     - redis
   build:
     context: ../
     dockerfile: ./docker/Dockerfile.backend
     target: production
   ports:
     - "30000:30000"
     - "31000:31000"
     - "32000:32000"
     - "18301:18301"
     - "18302:18302"
     - "8843:8843"
   networks:
     - app_network
 frontend:
   build:
     context: ../
     dockerfile: ./docker/Dockerfile.frontend
     target: production
   container_name: "${COMPOSE_PROJECT_NAME}_frontend"
   ports:
     - "80:80"
   networks:
     - app_network
 mongo:
   image: mongo:7.0
   container_name: "${COMPOSE_PROJECT_NAME}_mongo"
   env_file:
     - ../.env.mongo
   ports:
```

```
    - "27017:27017"
  restart: on-failure
  volumes:
    - $PWD/docker/mongo:/var/lib/mongo/data
  entrypoint: >
    /bin/bash -c '
    openssl rand -base64 756 > /data/keyfile &&
    chmod 400 /data/keyfile &&
    chown mongodb:mongodb /data/keyfile &&
    /usr/local/bin/docker-entrypoint.sh mongod --replSet rs0 --keyFile
/data/keyfile --bind_ip_all --auth'
  healthcheck:
    test: >
      mongosh
      -u ${MONGO_INITDB_ROOT_USERNAME}
      -p ${MONGO_INITDB_ROOT_PASSWORD}
      --eval "try { rs.status().ok } catch (e) { rs.initiate({ _id: 'rs0',
members: [{ _id: 0, host: 'localhost:27017' }] }).ok }"
    start_period: 0s
    interval: 500ms
    timeout: 5s
    retries: 5
  networks:
    - app_network
 redis:
  image: "redis:alpine"
  container_name: "${COMPOSE_PROJECT_NAME}_redis"
  command: "redis-server --appendonly yes"
  restart: unless-stopped
  ports:
    - "6379:6379"
  networks:
    - app_network

networks:
 app_network:
```

Отдельно заведем докерфайлы для сборки фронтенда и бекенда:

```
# Stage #1 Build backend
FROM node:18-alpine as builder

WORKDIR /app

COPY . .
```

```
RUN npm install

WORKDIR /app/backend

RUN npm install

RUN npm run build

# Stage #2 Run backend
FROM node:18-alpine as production

WORKDIR /app

# Install pm2-runtime
RUN npm install pm2 -g

COPY /backend/.env .
COPY /backend/package.json .
COPY /backend/ecosystem.config.js .
COPY /backend/game-bots ./game-bots

COPY --from=builder /app/backend/dist/. ./
COPY --from=builder /app/backend/node_modules ./node_modules
# Merge top level node_modules & backend node_modules
COPY --from=builder /app/node_modules/. ./node_modules

ENV NODE_ENV=production

CMD ["pm2-runtime", "ecosystem.config.js", "--env","production"]
```

```
# Stage #1 Build frontend
FROM node:18-alpine as builder

WORKDIR /app

COPY . .

RUN npm install

WORKDIR /app/frontend

RUN npm install --legacy-peer-deps
```

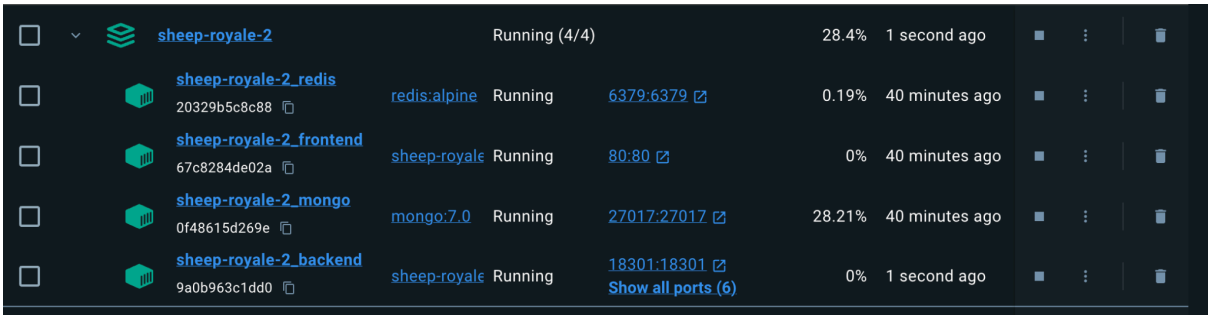
```
RUN npm run build

# Stage #2 Run frontend
FROM nginx:alpine-slim as production

WORKDIR /app

COPY --from=builder /app/frontend/dist/ /usr/share/nginx/html/.
COPY ./docker/nginx.conf /etc/nginx/conf.d/default.conf

CMD ["nginx", "-g", "daemon off;"]
```

Также подготовим конфиг для nginx, чтобы микросервисы бэкенда в докере корректно масшрутизировались.

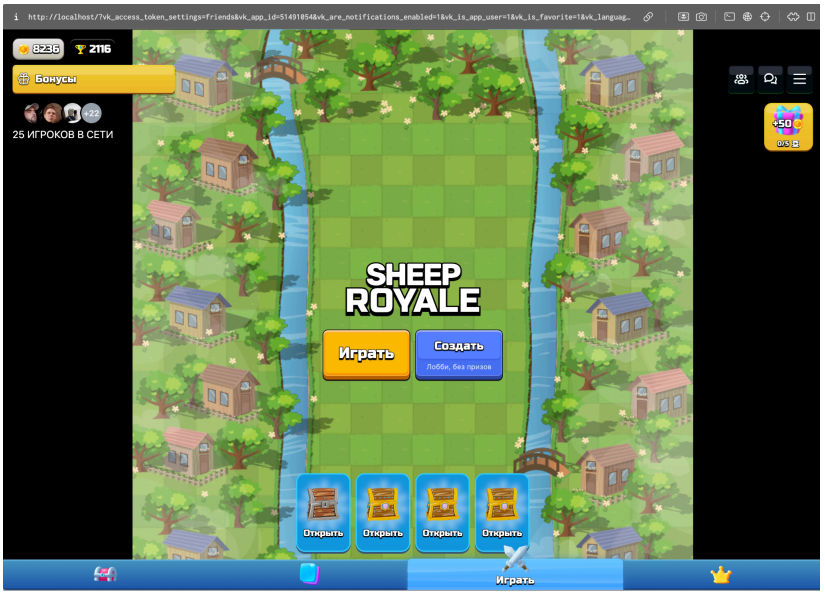
```
map $http_upgrade $connection_upgrade {
   default upgrade;
   ''      close;
}


server {
 listen 80;

 proxy_redirect off;
 proxy_set_header Host $host;
 proxy_set_header X-Real-IP $remote_addr;
 proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
 proxy_set_header X-Forwarded-Proto $scheme;
 proxy_set_header Upgrade $http_upgrade;
 proxy_set_header Connection $connection_upgrade;
 proxy_read_timeout 1m;
 proxy_connect_timeout 1m;
 client_max_body_size 20M;

 location / {
   root /usr/share/nginx/html/;
   try_files $uri $uri/ /index.html;
 }

 location /hub/ {
   proxy_pass http://backend:30000/;
 }

 location /waitroom/ {
```

```
    proxy_pass http://backend:31000/;
  }

  location /lobby/ {
    proxy_pass http://backend:32000/;
  }

  location /callback/ {
    proxy_pass http://backend:18301/;
  }

  location /notifications/ {
    proxy_pass http://backend:18302/;
  }
}
```

В результате, получим докер контейнер проекта содержащий в себе необходимые базы данных, фронтенд и бекенд проекта.



Как результат, получим рабочий проект собирающийся и запускающийся через докер.



Вывод

В ходе работы был настроен docker контейнер, который упростит дальнейшую развертку и работу с проектом. А также решит все проблемы с запуском проекта на различных OS.