Dockerização da aplicação Multiplayer-Soccer

Este guia mostra como executar o projeto em um container Docker.

1. Dockerfile

Crie um arquivo chamado `Dockerfile` na raiz do projeto (ao lado de `server.js`) com o seguinte conteúdo:

```
# Imagem base oficial do Node (LTS)
FROM node:20-alpine

# Diretório de trabalho dentro do container
WORKDIR /app

# Copia apenas package* primeiro para aproveitar cache
COPY package*.json ./

# Instala dependências em modo produção
RUN npm install --only=production

# Copia o restante do código
COPY . .

# Variável de ambiente da porta (o servidor usa process.env.PORT)
ENV PORT=3000

# Expõe a porta (o mapeamento real é feito no docker run)
EXPOSE 3000

# Comando de inicialização
CMD ["node", "server.js"]
```

Se você preferir iniciar pelo script `npm start`, altere a última linha para:

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

2. .dockerignore

Crie um arquivo `.dockerignore` na raiz do projeto para evitar copiar arquivos desnecessários para a imagem:

```
node_modules
npm-debug.log
Dockerfile
.dockerignore
.git
.gitignore
ngrok.yml
```

3. Construindo a imagem

No terminal, dentro da pasta do projeto `Multiplayer-Soccer`:

```
docker build -t multiplayer-soccer .
```

- `-t multiplayer-soccer` define o nome da imagem.

4. Executando o container

Para rodar o container mapeando a porta 3000 do container para a 3000 da máquina host:

```
docker run --rm --name multiplayer-soccer-container -p 3000:3000 multiplayer-soccer
```

Explicando o comando

- `docker run`: cria e inicia um novo container a partir de uma imagem.
- `--rm`: remove automaticamente o container quando ele for parado (útil em desenvolvimento para não acumular containers parados).
- `--name multiplayer-soccer-container`: define um nome legível para o container, facilitando o uso de comandos como `docker logs multiplayer-soccer-container` ou `docker stop multiplayer-soccer-container`.
- `-p 3000:3000`: faz o mapeamento de portas `host:container`.
  - Primeiro `3000`: porta da máquina host (onde você acessa pelo navegador).
  - Segundo `3000`: porta exposta dentro do container, onde o Node está escutando (`process.env.PORT` ou `3000`).

    - Resultado: acessar `http://localhost:3000` atinge o servidor que está rodando dentro do container.
- `multiplayer-soccer`: nome da imagem construída no passo 3 (`docker build -t multiplayer-soccer .`).

Acesse no navegador:

- `http://localhost:3000`

5. Usando outra porta no host (opcional)

Se a porta 3000 já estiver em uso na máquina host, você pode usar outra porta, por exemplo 8080:

```
docker run --rm -p 8080:3000 multiplayer-soccer
```

E acessar:

- `http://localhost:8080`

6. Deploy em AWS EC2 com Docker e Elastic IP

Este é um fluxo simples para rodar a aplicação conteinerizada em uma instância EC2 acessível por um Elastic IP.

6.1 Criar instância EC2 e Elastic IP

1. No console da AWS, crie uma instância EC2 (por exemplo Amazon Linux 2 ou Ubuntu).
2. Em "Elastic IPs", aloque um novo Elastic IP.
3. Associe o Elastic IP à instância EC2 criada.

Depois disso, você terá um IP público fixo, algo como `X.X.X.X`, que usará para acessar a aplicação.

6.2 Conectar na EC2 e instalar Docker

Conecte na instância via SSH, por exemplo (ajuste usuário/arquivo `.pem` conforme sua AMI):

```
ssh -i /caminho/sua-chave.pem ec2-user@X.X.X.X
```

No Amazon Linux 2, instale e inicie o Docker:

```
sudo yum update -y
sudo yum install -y docker
sudo service docker start
sudo usermod -aG docker ec2-user
```

Saia e entre novamente na sessão SSH para aplicar o grupo `docker`:

```
exit
ssh -i /caminho/sua-chave.pem ec2-user@X.X.X.X
```

Teste se o Docker está ok:

```
docker ps
```

6.3 Enviar o projeto para a EC2

Na sua máquina local, dentro da pasta do projeto `Multiplayer-Soccer`:

```
scp -i /caminho/sua-chave.pem -r . ec2-user@X.X.X.X:~/Multiplayer-Soccer
```

Na EC2:

```
cd ~/Multiplayer-Soccer
```

6.4 Buildar e rodar o container na EC2

Build da imagem na instância:

```
docker build -t multiplayer-soccer .
```

Rodar o container em background expondo a porta 3000:

```
docker run -d --rm --name multiplayer-soccer-container -p 3000:3000 multiplayer-
soccer
```

- `-d`: roda o container em segundo plano (detached).
- `--rm`: remove o container quando ele for parado.
- `--name multiplayer-soccer-container`: nome legível para o container.
- `-p 3000:3000`: mapeia porta 3000 da EC2 → porta 3000 do container.

6.5 Liberar a porta no Security Group

No console da AWS:

1. Vá em EC2 → Instâncias → selecione sua instância.
2. Na aba "Security", clique no Security Group associado.
3. Em "Inbound rules", adicione uma regra:
    - Type: `Custom TCP`
    - Port: `3000`
    - Source: `0.0.0.0/0` (para testes; depois restrinja conforme necessário).

6.6 Acessar pelo Elastic IP

No navegador, acesse:

- `http://X.X.X.X:3000`

Se preferir expor na porta 80 (HTTP padrão), rode o container assim:

```
docker run -d --rm --name multiplayer-soccer-container -p 80:3000 multiplayer-soccer
```

E então:

1. Abra a porta 80 no Security Group (Type: HTTP ou Custom TCP 80).

2. Acesse apenas:

- `http://X.X.X.X`

onde `X.X.X.X` é o seu Elastic IP.

7. Portar a imagem para outra máquina (Windows/Linux/macOS)

Existem duas formas principais de rodar esta aplicação em outra máquina que tenha Docker instalado.

7.1 Opção 1 – Levar apenas o código e rebuildar

Na outra máquina (Windows, Linux ou macOS), com Docker já instalado:

1. Copie o projeto (via `git clone`, ZIP, pendrive etc.).
2. Dentro da pasta do projeto `Multiplayer-Soccer`, construa a imagem:

```
docker build -t multiplayer-soccer .
```

3. Rode o container normalmente:

```
docker run -d --rm --name multiplayer-soccer-container -p 3000:3000 multiplayer-soccer
```

4. Acesse na máquina de destino:
    - `http://localhost:3000`

Se quiser acessar de outro dispositivo na mesma rede, use o IP da máquina de destino, por exemplo:

- `http://IP_DA_MAQUINA:3000`

7.2 Opção 2 – Exportar e importar a imagem pronta

Se você não quiser rebuildar a imagem em cada máquina, pode exportar a imagem pronta e importar em outra máquina.

Na máquina de origem (onde a imagem já existe)

1. Salve a imagem em um arquivo `.tar`:

```
docker save -o multiplayer-soccer.tar multiplayer-soccer
```

2. Transfira o arquivo `multiplayer-soccer.tar` para a outra máquina
    - Pode ser via pendrive, compartilhamento de rede, `scp`, etc.
    - Exemplo usando `scp`:

```
scp multiplayer-soccer.tar usuario@OUTRO_IP:/caminho/destino/
```

Na máquina de destino

3. Importe a imagem a partir do arquivo `.tar`:

   ```
   docker load -i /caminho/destino/multiplayer-soccer.tar
   ```

   Após esse comando, `docker images` deve listar a imagem `multiplayer-soccer`.

4. Rode o container normalmente:

   ```
   docker run -d --rm --name multiplayer-soccer-container -p 3000:3000 multiplayer-soccer
   ```

5. Acesse na máquina de destino:

   - `http://localhost:3000`

Ou, a partir de outra máquina na rede:

- `http://IP_DA_MAQUINA:3000`