VPWA: Cvičenie DevOps, 1. časť - Docker

Ciele cvičenia:

- Precvičiť si praktickú prácu s platformou Docker.
- Využiť Docker na kontajnerizáciu (tzv. "dockerizáciu") vlastnej PWA na báze AdonisJS (BE) + QuasarJS (FE).

Prerekvizity:

- Nainštalovaná platforma Docker so súčasťou Docker Compose.

Návod na inštaláciu (MS Windows / macOS):

1. Stiahnite a nainštalujte si distribúciu Docker for Desktop pre váš systém.
2. V prípade systému MS Windows je odporúčaný backend WSL 2 (voľba v inštalátore).

Návod na inštaláciu (Ubuntu a systémy s apt):

```
sudo apt update
```

```
sudo apt install -y docker.io docker-compose
```

Úloha č. 1: Dockerizujte vašu progresívnu aplikáciu (FE časť slek-client):

Zostavte Docker image umožňujúci nasadenie **FE časti vašej PWA ako mikroslužby**. Využite možnosť **preťažiť premenné prostredia počas buildu (build-stage)** - najmä URL pre API server (direktíva **ARG** - pozrite Docker Cheat Sheet), ako aj **premenné prostredia servera vo výslednom kontajneri (production-stage)** (direktíva **ENV**) - napr. HOST, PORT. Výsledný Docker image následne spustite spolu s lokálne rozbehaným API serverom (slek-server) a skontrolujte, či vaša aplikácia funguje korektne.

Ako príklad môžete použiť nasledovný Dockerfile (uložte ho do priečinka /slek-client):

```
```

```
# ----- BUILD STAGE -----
FROM node:lts as build-stage

# Aliases setup for container folders
ARG PWA_src="."
ARG DIST="/pwa"

# Define arguments which can be overridden at build time
ARG API_URL="https://prod.api.com"

# Set the working directory inside the container to server module
WORKDIR ${DIST}

# Copying in two separate steps allows us to take advantage of cached Docker layers.
COPY ${PWA_src}/package*.json ./

# Install dependencies
RUN npm install

# Copy source files inside container
COPY ${PWA_src} .

# Build the SPA
RUN npx @quasar/cli build -m pwa

# ----- PRODUCTION STAGE -----
FROM node:lts as production-stage

# Aliases setup for container folders
ARG DIST="/pwa"
ARG PWA="/myapp"

# Define environment variables for HTTP server
ENV HOST="0.0.0.0"
ENV PORT="8080"

# Set working directory
WORKDIR ${PWA}

# Copy build artifacts from previous stage
```

```
COPY --from=build-stage ${DIST}/dist/pwa ./

# Expose port outside container
EXPOSE ${PORT}

# Install pm2
RUN npm install -g @quasar/cli

# Start server module inside the container
CMD ["quasar", "serve"]
```

Spustite build podľa novovytvoreného Dockerfile (z priečinka slek-client):

```
docker build -f Dockerfile --build-arg API_URL=http://localhost:3333 --tag=slek-client-image .
```

Spustite slek-client z **vytvoreného Docker image**:

```
docker run --rm -p 8080:8080 slek-client-image
```

Úloha č. 2: Dockerizujte vašu progresívnu aplikáciu (BE časť slek-server):

Zostavte Docker image umožňujúci nasadenie **BE časti vašej PWA ako mikroslužby**. Definujte v rámci Dockerfile predvolené hodnoty pre premenné prostredia (direktíva ENV). Pre jednoduchosť môžete použíť single-stage build. Vytvorte Docker image tak, aby bol AdonisJS server zostavený a spustený v **režime production** (pomôcka). Nezabudnite v rámci Dockerfile **inicializovať Sqlite**. Výsledný Docker image spustite. Následne spustite aj Docker image slek-klient (FE časť) a skontrolujte funkcionalitu aplikácie.

Ako príklad môžete použiť nasledovný Dockerfile:

```
# Include the latest node image
FROM node:lts

# Aliases setup for container folders
ARG SERVER="/slek-server"
ARG SERVER_src="."
ARG BUILD="/slek-server/build"

# Define environment variables for server (see .env)
```

```
ENV HOST=0.0.0.0
ENV PORT=3333
ENV NODE_ENV=development
ENV APP_KEY=nlnpGYSTleLKKrMtkZSPJfI8tHJWMIa9
ENV DRIVE_DISK=local
ENV DB_CONNECTION=sqlite

ENV PORTS="3333"

# Set the working directory inside the container to server module
WORKDIR ${SERVER}

# Expose port outside container
EXPOSE ${PORTS}

# Copy server module
COPY ${SERVER_src} ${SERVER}

# Build TS files
RUN node ace build --production

# Update workdir
WORKDIR ${BUILD}

# Install production dependencies
RUN npm ci --production

# Initialize Sqlite
RUN mkdir tmp && touch tmp/db.sqlite3

# Migrate and seed database
RUN node ace migration:refresh --seed

# Start server module inside the container
CMD ["node", "server.js"]
```

Vykonajte **build** a **spustite** vytvorený Docker image. Následne **preverte funkcionalitu aplikácie** otvorením nasledovnej adresy v prehliadači (alebo analogicky pre váš prípad):

```
http://localhost:8080
```

(Voliteľné) Zabezpečte **perzistenciu dát** pomocou Docker volume pre váš slek-server. Pomôcka: pozrite Docker cheat sheet.

Revision #18
Created Mon, Nov 21, 2022 12:31 PM by Adam Puskas
Updated Mon, Nov 21, 2022 11:31 AM by Adam Puskas