VPWA: Cvičenie DevOps, 2. časť - Kompozícia služieb, nasadenie v cloude (AWS)

Ciele cvičenia

- Oboznámiť sa s nasadzovaním kontajnerizovanej aplikácie na platforme AWS
  - AWS ECS (orchestrácia) + AWS EC2 (server)
- Pripraviť sa na nasadenie vlastnej "dockerizovanej" aplikácie
  - Server (AdonisJS), Klient (QuasarJS), RDBMS databáza (Postgres, MariaDB, MySQL)
- Vybrať si cloud platformu (AWS, Digital Ocean, MS Azure, Google Cloud), zaregistrovať sa a oboznámiť sa s jej prostredím, službami.

Prerekvizity

- Zostavené a funkčné Docker images server + PWA klient, pripravené na nasadenie
  - Konfigurovateľné premenné prostredia (ENV)
  - Zabezpečená perzistencia dát (RDBMS) - súčasť dnešného cvičenia

Návod na získanie kreditov pre Digital Ocean, MS Azure (GitHub Student Developer Pack)

1. Zaregistrujte si GitHub Student Developer Pack, využite univerzitnú @stuba mailovú adresu.
2. Dokončite registráciu na Digital Ocean (alebo MS Azure) a prihláste sa do služby.

Úloha č. 1: Pripravte svoju aplikáciu na cloudové nasadenie

Zostavte kompozíciu svojej PWA pomocou nástroja Docker Compose. Zabezpečte zostavenie (build) jednotlivých Docker images (server, klient) v produkčnom móde a vystavte jednotlivé premenné prostredia (ENV) v docker-compose.yml pre jednoduchú konfiguráciu. Zabezpečte perzistenciu dát prostredníctvom vhodnej RDBMS (odporúča sa Postgres).

- Výsledná kompozícia pozostáva zo:
  - server časti (slek-server)
  - klient časti (slek-klient)

- SQL databázy (napr. Postgres)
- Premenné prostredia sú konfigurovateľné priamo cez docker-compose.yml (ENV: HOST, PORT, API_URL...)
- Funkčný produkčný build pre server aj klient časť

Pre slek-server môžete použiť nasledovný ukážkový multi-stage Dockerfile (upravte si ho podľa potreby):

```
# ----- BUILD STAGE -----
# Include the latest node image
FROM node:lts as build-stage
# Aliases setup for container folders
ARG SERVER="/slek-server"
ARG SERVER_src="."
ARG BUILD="/slek-server/build"
# Set the working directory inside the container to server module
WORKDIR ${SERVER}
# Copy server module
COPY ${SERVER_src} ${SERVER}
# Build dependencies and TS files
RUN npm i
RUN node ace build --production

# ----- PRODUCTION STAGE -----
FROM node:lts as production-stage
# Aliases setup
ARG BUILD="/slek-server/build"
ARG SERVER="/myserver"
# Define environment variables for server (see .env)
ENV HOST=0.0.0.0
ENV PORT=3333
ENV NODE_ENV=production
ENV APP_KEY=nlnpGYSTleLKKrMtkZSPJfI8tHJWMIa9
ENV DRIVE_DISK=local
ENV DB_CONNECTION=pg
ENV DB_HOST=slek-postgres
ENV DB_PORT=5432
ENV DB_USER=slek
ENV DB_PASSWORD=slekpass
ENV DB_NAME=slek
# Set workdir
```

```
WORKDIR ${SERVER}
# Copy build artifacts from previous stage
COPY --from=build-stage ${BUILD} ./
# Install production dependencies
RUN npm ci --production
# Expose port outside container
EXPOSE ${PORT}
# Start server module inside the container
CMD ["node", "server.js"]
```

Dockerfile pre slek-client môžete použiť z predošlého cvičenia. Nezabudnite do slek-server kódu pridať konfiguráciu pre Postgres driver. Pomôcka (config/database.ts):

```
    pg: {
      client: 'pg',
      connection: {
        host: Env.get('DB_HOST'),
        port: Env.get('DB_PORT'),
        user: Env.get('DB_USER'),
        password: Env.get('DB_PASSWORD', ''),
        database: Env.get('DB_NAME'),
      },
      migrations: {
        naturalSort: true,
      },
      healthCheck: false,
      debug: false,
    },
```

Taktiež je potrebné pridať príslušný driver (modul) do dependencies v package.json. Pomôcka:

```
"pg": "^8.7.3"
```

Vytvorte súbor docker-compose.yml s inštrukciami pre build a spustenie kompozície modulov vašej PWA, vrátane databázy. Príklad:

```
version: '3.5'
services:
  slek-client:
    image: slek-client-local:latest
    # Path to slek-client repository
```

```
    build: ./slek-client
    restart: unless-stopped
    container_name: slek-client-local
    ports:
      - 8080:8080
    environment:
      HOST: 0.0.0.0
      PORT: 8080
      NODE_ENV: production

  slek-server:
    image: slek-server:latest
    build: ./slek-server
    restart: unless-stopped
    container_name: slek-server
    ports:
      - 3333:3333
    environment:
      HOST: 0.0.0.0
      PORT: 3333
      APP_KEY: 7QSD22E4g2i9KQ9XDm5KPDJQN6CjuUY5YQnwfaRPzz
      DRIVE_DISK: local
      NODE_ENV: production
      DB_CONNECTION: pg
      DB_HOST: slek-postgres
      DB_PORT: 5432
      DB_USER: slek
      DB_PASSWORD: slekpass
      DB_NAME: slek

  slek-postgres:
    image: postgres
    restart: unless-stopped
    ports:
      - "5432:5432"
    container_name: "slek-postgres"
    volumes:
      - postgres:/var/lib/postgresql/data
    environment:
      POSTGRES_USER: "slek"
```

```
      POSTGRES_PASSWORD: "slekpass"
      POSTGRES_DB: "slek"

volumes:
  postgres:
    name: pg-volume
```

Spustite build pomocou:

```
docker-compose -f docker-compose.yml build
```

Spustite kompozíciu:

```
docker-compose -f docker-compose.yml up
```

Vykonajte seednutie databázy a preverte funkcionalitu vašej aplikácie:

```
docker exec -it slek-server /bin/bash
node ace migration:refresh --seed
```

Úloha č. 2: Oboznámte sa s nasadzovaním na platforme AWS ECS (Elastic Container Service)

V spolupráci s cvičiacim sa oboznámte s nasadzovaním kontajnerizovanej webovej aplikácie (klient, server, RDBMS databáza) na platformu AWS ECS.

- Vytvorenie ECS klastra (EC2 inštancia t3.micro + podporné služby)
- Definícia úlohy (task definition)
- Vytvorenie Docker registra (ECR)
- Vytvorenie služby (ECS service)
- Nastavenie siete a bezpečnostných skupín (security groups)

Následne si vyberte niektorú cloud platformu (AWS, Digital Ocean, MS Azure, Google Cloud), zaregistujte sa v nej sa a oboznámte sa s jej prostredím a službami (registrácia je prerekvizitou pre nasledujúce cvičenie).

Revision #10
Created Sun, Nov 27, 2022 8:48 PM by Adam Puskas
Updated Sun, Nov 27, 2022 7:48 PM by Adam Puskas