Pràctica Docker

Jairo Tecles

En la carpeta arrel del vostre codi, haurà d'existir un fitxer docker-compose.yml que implementarà el següent:

- Servei de MySQL:
  - Utilitzarà la imatge docker oficial de MySQL
  - El contenidor associat es denominarà mysql_contenidor
  - Serà el primer servei a arrancar
  - La primera tasca que farà res més arrancar serà crear les taules necessàries (script en la tasca).
  - Has de crear un volum per a que no es perden les dades de la base de dades
  - Tots els fitxers de configuració necessaris residiran en una carpeta en el projecte

```yaml
mysql:
  image: mysql:8.0
  container_name: mysql_contenidor
  restart: unless-stopped
  env_file: [.env]
  ports:
    - 3306:3306
  volumes:
    - ./mysql_data:/var/lib/mysql
    - ./mysql-init:/docker-entrypoint-initdb.d
```

- El volum mysql_data guarda tota la informació de la base de dades.
- El volum mysql-init és el que guarda el script amb l'esquelet de la bbdd.
- El arxiu .env assignarà les variables d'entorn necessàries.

- Servei de backend:
  - S'ha de construir una imatge (Dockerfile) utilitzant multi-stage build, en la part final s'ha d'utilitzar una imatge d'alpine
  - No arrancarà fins que el servei de MySQL no estiga preparat completament (wait-for-it)
  - El contenidor associat es denominarà backend_contenidor
  - Executarà com a primer comando res més arrancar el comando necesari per posar en marcha el servidor

```yaml
backend:
  build: ./backend
  container_name: backend_contenidor
  restart: unless-stopped
  depends_on:
    - mysql
  ports:
    - 8000:8000
  env_file: [.env]
```

```dockerfile
FROM php:8.2-cli (last pushed 6 days ago)

RUN docker-php-ext-install pdo pdo_mysql

WORKDIR /var/www/html

COPY index.php config.php items.php ./

COPY wait-for-it.sh /usr/local/bin/wait-for-it.sh
RUN chmod +x /usr/local/bin/wait-for-it.sh

EXPOSE 8000

CMD ["sh", "-c", "wait-for-it.sh
mysql_contenidor:3306 -- php -S 0.0.0.0:8000
index.php"]
```

- Aquest servei utilitza una imatge personalitzada basada en php.
- Ja que el codi es comunica amb una base de dades, he instal·lat les dependències necessàries
- El contenidor esperarà a que la base de dades estiga online, ja que encara que en el docker-compoose.yml posem depends_on: mysql, la base de dades encara no està 100% operativa

- Servei de frontend:
  - Arrancarà després del servei de backend
  - El contenidor associat es denominarà frontend_contenidor
  - Executarà com a primer comando res més arrancar: npm run serve

```yaml
frontend:
  build: ./frontend
  container_name: frontend_contenidor
  restart: unless-stopped
  depends_on:
    - backend
  ports:
    - 8080:80
```

```dockerfile
FROM node:18 AS builder (last pushed 6 months ago)

WORKDIR /app

COPY . .

RUN npm i
RUN npm run build

FROM nginx:stable-alpine (last pushed 1 month ago)

COPY --from=builder /app/dist /usr/share/nginx/html

EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

- Aquest servei utilitza una imatge personalitzada basada en node amb multi stage.
- La primera stage, s'encarrega de construir la app, i la segona, la serveix utilitzant nginx
- Utilitzar 2 stages, ens ha proporcionat una image més lleugera, ja que sols ens quedem amb els arxius de la build.

- Servei phpmyadmin
  - Ens permetrà administrar la base de dades de MySQL
  - Utilitzarà la imatge docker oficial de phpmyadmin
  - El contenidor associat es denominarà adminMySQL_contenidor
  - Arrancarà després del servei de MySQL

```yaml
phpmyadmin:
  image: phpmyadmin/phpmyadmin
  container_name: adminMySQL_contenidor
  restart: unless-stopped
  depends_on:
    - mysql
  env_file: [.env]
  environment:
    PMA_HOST: mysql_contenidor
    PMA_USER: ${DB_USER}
    PMA_PASSWORD: ${DB_PASSWORD}

  ports:
    - 8001:80
```

- Arxius de interés:

```
.env
1  MYSQL_ROOT_PASSWORD=rootpassword
2  MYSQL_DATABASE=elementos
3  MYSQL_USER=usuario_db
4  MYSQL_PASSWORD=password_db
5
6  DB_HOST=mysql_contenidor
7  DB_NAME=elementos
8  DB_USER=usuario_db
9  DB_PASSWORD=password_db
```