Documentación Desafio 5 - NestJS + MongoDB con Docker

Introducción

Durante este desafío, desarrollé un entorno de desarrollo local utilizando Docker y Docker Compose para una aplicación NestJS junto a una base de datos MongoDB.

Contenido del Proyecto

El repositorio contiene los siguientes archivos principales:

- Dockerfile: Define la imagen de la aplicación NestJS.
- docker-compose.yaml: Orquesta los servicios de la aplicación y MongoDB.
- README.md: Documentación técnica y guía paso a paso.

Requisitos Previos

Antes de comenzar, me aseguré de tener instalados Docker y Docker Compose en mi sistema.

Si no los tenés instalados, podés seguir estos pasos:

Instalación en Linux (Ubuntu/Debian)

```bash
sudo apt update
sudo apt install -y docker.io docker-compose
sudo systemctl enable docker
sudo systemctl start docker
```

Instalación en macOS

1. Descargar Docker Desktop desde: https://www.docker.com/products/docker-desktop
2. Instalar y seguir las instrucciones del instalador.

Instalación en Windows

1. Descargar Docker Desktop desde: https://www.docker.com/products/docker-desktop

2. Activar WSL2 si se solicita durante la instalación.

3. Reiniciar y lanzar Docker Desktop.

Para verificar que Docker esté funcionando correctamente:

```bash
docker --version
docker-compose --version
```

Fork del Repositorio

Para trabajar sobre la base del proyecto, realicé un fork del repositorio original:

https://github.com/yosoyfunes/app-template-nestjs

Luego, cloné mi propia copia del repositorio con:

git clone https://github.com/mi-usuario/app-template-nestjs.git

cd app-template-nestjs

Después de realizar las modificaciones necesarias (Dockerfile, docker-compose.yaml y documentación), realicé el commit y push al repositorio forkeado:

git add .

git commit -m "feat: entorno Docker + documentación"

git push origin main

Creación del archivo Dockerfile

FROM node:18

WORKDIR /usr/src/app

COPY package*.json ./

```
RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "run", "start:dev"]
```

Creación del archivo docker-compose.yaml

```
version: "3.9"

services:
  app:
    build: .
    ports:
      - "3000:3000"
    volumes:
      - .:/usr/src/app
      - /usr/src/app/node_modules
    depends_on:
      - mongo
    environment:
      - MONGO_URI=mongodb://mongo:27017/nestdb
    command: npm run start:dev

  mongo:
    image: mongo:6
    restart: always
    ports:
      - "27017:27017"
```

```
    volumes:
      - mongo-data:/data/db

volumes:
  mongo-data:
```

Construcción y levantamiento de contenedores

Ejecuté el siguiente comando para construir las imágenes y levantar los servicios:

docker compose up -d --build

Verificación del Entorno

Verifiqué que los contenedores estuvieran corriendo correctamente utilizando:

docker ps

Finalmente, abrí el navegador y confirmé que la aplicación respondía en:

http://localhost:3000

Configuración Utilizada

La variable de entorno de conexión a MongoDB fue definida dentro del archivo docker-compose.yaml como:

MONGO_URI=mongodb://mongo:27017/nestdb

Utilicé un volumen llamado 'mongo-data' para mantener la persistencia de la base de datos.

Evidencia de Funcionamiento

Capturé capturas de pantalla que demuestran:

- El comando 'docker compose up -d' ejecutado correctamente.
- El navegador accediendo a http://localhost:3000.
- La salida de 'docker ps' mostrando ambos contenedores activos.

Cierre de Contenedores

Para detener y eliminar los contenedores, utilicé el comando:

docker compose down

Conclusión

Este entorno permitirá a cualquier desarrollador del equipo levantar la aplicación con un solo comando, replicando un entorno productivo de desarrollo.