

DESAFIO N10

Acosta Hernan Andres (993394)

Para cumplimentar con lo requerido en el desafío N10 se procedió de la siguiente manera:

Requisitos:

1. Elaborar el archivo Dockerfile con todas las instrucciones necesarias para utilizar la aplicación.
2. Entregar un archivo docker-compose.yaml que permita al desarrollador levantar un entorno de trabajo local con un simple comando.
3. Elaborar toda la documentación necesaria.

A continuación se proporcionaran los archivos Dockerfile y docker-compose

1. Dockerfile

El archivo Dockerfile define cómo se construye la imagen Docker de la aplicación.

Aquí está el contenido:

```
# Utilizar la imagen base de Node.js (basada en la versión de Node.js en package.json)
```

```
FROM node:20-alpine AS builder
```

```
# Establecer el directorio de trabajo dentro del contenedor
```

```
WORKDIR /app
```

```
# Copiar los archivos package.json y package-lock.json (o yarn.lock)
```

```
COPY package*.json ./
```

```
# Instalar las dependencias
```

```
RUN npm install
```

```
# Copiar el código fuente de la aplicación
```

```
COPY ..
```

```
# Construir la aplicación NestJS
```

```
RUN npm run build
```

```
# Stage 2: Production image
```

```
FROM node:20-alpine
```

```
WORKDIR /app
```

```
# Copiar los archivos contruidos del stage anterior
```

```
COPY --from=builder /app/dist ./dist
```

```
COPY --from=builder /app/node_modules ./node_modules
```

```
COPY --from=builder /app/package*.json ./
```

```
# Exponer el puerto en el que la aplicación escucha
EXPOSE 3000
```

```
# Comando para iniciar la aplicación
CMD [ "node", "dist/main" ]
```

Explicación:

- Stage 1 (Builder):
 - Usa una imagen de Node.js para instalar dependencias y construir la aplicación.
 - Copia los archivos necesarios (package.json, package-lock.json, y el código fuente).
 - Ejecuta npm run build para compilar la aplicación TypeScript a JavaScript.
- Stage 2 (Producción):
 - Usa una imagen ligera de Node.js para la ejecución.
 - Copia solo los archivos necesarios desde el stage anterior (dist, node_modules, y package.json).
 - Expone el puerto 3000 y ejecuta la aplicación con node dist/main.

2. docker-compose.yml

El archivo docker-compose.yml define los servicios necesarios para levantar el entorno local, incluyendo la aplicación y MongoDB.

```
version: "3.8"
services:
  app:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    depends_on:
      - mongo
    environment:
      - PORT=3000 # O el puerto que desees
      - MONGO_DB_URI=mongodb://mongo:27017
      - MONGO_DB_NAME=<db_name_here> # Reemplaza con el nombre de tu base de datos
      - MONGO_DB_USER=<mongo_user_here> # Reemplaza con el usuario de MongoDB
      - MONGO_DB_PASS=<mongo_pass_here> # Reemplaza con la contraseña de MongoDB
    restart: always

  mongo:
```

```
image: mongo:latest
ports:
- "27018:27017"
environment:
MONGO_INITDB_ROOT_USERNAME: <mongo_user_here> # Reemplaza con el usuario root de
MongoDB
MONGO_INITDB_ROOT_PASSWORD: <mongo_pass_here> # Reemplaza con la contraseña root de
MongoDB
volumes:
- mongo_data:/data/db
restart: always
```

```
volumes:
mongo_data:
```

Explicación:

- Servicio app:
 - Construye la aplicación usando el Dockerfile.
 - Expone el puerto 3000 para acceder a la aplicación.
 - Depende del servicio mongo para la base de datos.
 - Configura las variables de entorno necesarias.
- Servicio mongo:
 - Usa la imagen oficial de MongoDB.
 - Expone el puerto 27018 para acceder a MongoDB desde el host local.
 - Configura las credenciales de MongoDB.
 - Usa un volumen para persistir los datos de la base de datos.

README.md

```
# Nombre del Proyecto
# educacionit-app
```

Breve descripción

Aplicación NestJS con MongoDB.

Tecnologías Utilizadas

- **NestJS**: Framework de Node.js para construir aplicaciones escalables.
- **Node.js**: Entorno de ejecución para JavaScript.
- **TypeScript**: Lenguaje principal para el desarrollo.
- **Docker**: Contenerización de la aplicación para facilitar el despliegue.
- **MongoDB**: Base de datos NoSQL utilizada para almacenar datos.

Requisitos Previos

- **Docker**: Asegúrate de tener Docker instalado en tu máquina.
- **Docker Compose**: Asegúrate de tener Docker Compose instalado.

Instalación

2. Navega al directorio del proyecto:

```
``bash
cd nombre-del-proyecto
```

hernan@andres:~/Educacionit-app\$ cd nestjs-docker

Cómo Usar la Aplicación

Levantar el Entorno con Docker Compose

1. Construye y levanta los contenedores:

```
docker-compose up --build
```

detener el Entorno con Docker Compose

```
docker-compose down
```

Este comando detendrá y eliminará los contenedores, redes y volúmenes definidos en tu archivo docker-compose.yml.

Si solo quieres detener los contenedores sin eliminarlos, puedes usar:

```
docker-compose stop
```

Esto detendrá los contenedores, pero no los eliminará, lo que te permitirá reiniciarlos más tarde con docker-compose start.

2. Accede a la aplicación:

La aplicación estará disponible en <http://localhost:3000/>.

3. Verifica los logs:

Puedes ver los logs de la aplicación y MongoDB en la terminal donde ejecutaste docker-compose.

Endpoints Disponibles

- GET /:

- Descripción: Devuelve un mensaje de "Hello World!".

- Ejemplo

```
curl -X GET http://localhost:3000/
```

- Respuesta

"Hello World!"

```
hernan@andres:~/Educacionit-app/nestjs-docker$ curl -X GET http://localhost:3000/
Hello World!hernan@andres:~/Educacionit-app/nestjs-docker$
```

Interacción con MongoDB

Si has configurado MongoDB correctamente, la aplicación se conectará automáticamente a la base de datos. Puedes verificar la conexión revisando los logs de la aplicación.

```
hernan@andres:~/Educacionit-app/nestjs-docker$ sudo systemctl status mongod
mongod.service - MongoDB Database Server
Loaded: loaded (/usr/lib/systemd/system/mongod.service; disabled; preset: enabled)
Active: active (running) since Sat 2025-03-08 23:24:35 -03; 3s ago
Docs: https://docs.mongodb.org/manual
Main PID: 33432 (mongod)
Memory: 300.1M (peak: 365.9M)
CPU: 1.383s
CGroup: /system.slice/mongod.service
└─33432 /usr/bin/mongod --config /etc/mongod.conf

mar 08 23:24:35 andres systemd[1]: Started mongod.service - MongoDB Database Server.
mar 08 23:24:35 andres mongod[33432]: {"t":{"$date":"2025-03-09T02:24:35.916Z"},"s":"I", "c":"CONTROL", "id":7484500, "ctx":"main",
lines 1-12/12 (END)
```

Detener el Entorno

Para detener los contenedores, ejecuta:

docker-compose down

Estructura del Proyecto

- src/: Contiene el código fuente de la aplicación.
 - app.controller.ts: Controlador principal.
 - app.service.ts: Servicio principal.
 - app.module.ts: Módulo principal.
 - main.ts: Punto de entrada de la aplicación.
- test/: Pruebas unitarias y de integración.
- Dockerfile: Configuración de Docker para la aplicación.
- docker-compose.yml: Configuración de servicios Docker (aplicación y MongoDB).

Comando docker ps

```
hernan@andres:~/Educacionit-app/nestjs-docker$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
20b22f4683d1   mongo:latest   "docker-entrypoint.s..." 15 minutes ago Up 15 minutes 27017/tcp              magical_swartz
bd8484b1fb5c   nestjs-docker-app-1   "docker-entrypoint.s..." 3 hours ago   Up 2 hours    0.0.0.0:3000->3000/tcp   nestjs-docker-
app-1
282fad4c5478   mongo:latest   "docker-entrypoint.s..." 3 hours ago   Up 3 hours    0.0.0.0:27018->27017/tcp nestjs-docker-
mongo-1
```

Verifica la conexión a MongoDB:

Conéctate a la instancia de MongoDB desde tu máquina local usando mongosh:
mongosh "mongodb://localhost:27018"

```
hernan@andres:~/Educacionit-app/nestjs-docker$ mongosh "mongodb://localhost:27018"
Current Mongosh Log ID: 67cd0154038a5f0a0f6b140a
Connecting to:      mongodb://localhost:27018/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.4.2
Using MongoDB:      8.0.5
Using Mongosh:      2.4.2

For mongosh info see: https://www.mongodb.com/docs/mongosh-shell/
```

Accede a la aplicación en <http://localhost:3000>.



Con estos pasos quedo demostrado que el entorno de desarrollo es funcional y reproducible en cualquier maquina local, lo que facilita la colaboración entre miembros del equipo.

Conclusión