

# Desafío 10

## Bootcamp Devops Engineer

### Desarrollo:

Vamos a crear dos archivos principales:

1. Dockerfile: Este archivo contendrá las instrucciones para construir la imagen Docker de la aplicación NestJS.
2. docker-compose.yaml: Este archivo definirá los servicios necesarios, incluyendo la aplicación y la base de datos MongoDB, y gestionará su configuración y conexión.

- **Dockefile.**

Este es el archivo que Docker utilizará para construir la imagen de la aplicación NestJS. Asegúrate de tener el archivo Dockerfile en el directorio raíz del proyecto.

```
Dockerfile
1  # Usa la imagen oficial de Node.js como base
2  FROM node:18
3
4  # Establece el directorio de trabajo en el contenedor
5  WORKDIR /usr/src/app
6
7  # Copia el archivo package.json y package-lock.json al contenedor
8  COPY package*.json ./
9
10 # Instala las dependencias de la aplicación
11 RUN npm install
12
13 # Copia el resto del código de la aplicación al contenedor
14 COPY . .
15
16 # Expone el puerto en el que la aplicación va a correr
17 EXPOSE 3000
18
19 # Ejecuta el comando para iniciar la aplicación
20 CMD ["npm", "run", "start:dev"]
```

- **docker-compose.yaml**

El archivo docker-compose.yaml orquesta la ejecución de la aplicación y MongoDB en contenedores separados.

```
docker-compose.yaml
1  version: '3.8'
2  services:
3    mongodb:
4      image: mongo:latest
5      container_name: mongodb
6      ports:
7        - "27017:27017"
8      environment:
9        MONGO_INITDB_ROOT_USERNAME: root
10       MONGO_INITDB_ROOT_PASSWORD: password
11
12     volumes:
13       - mongo-data:/data/db
14
15     app:
16       container_name: desafio10-app
17       build: .
18       ports:
19         - "3000:3000"
20
21       environment:
22         MONGO_DB_URI: mongodb://mongodb:27017
23         MONGO_DB_NAME: app-desafio10
24         MONGO_DB_USER: root
25         MONGO_DB_PASS: password
26       depends_on:
27         - mongodb
28
29     volumes:
30       mongo-data:
```

- **Construir y Levantar los Contenedores**

Ejecutar el comando:

`docker-compose up --build -d`

```
[+] Building 7.3s (11/11) FINISHED
=> [app internal] load build definition from Dockerfile
=> => transferring dockerfile: 594B
=> [app internal] load metadata for docker.io/library/node:18
=> [app auth] library/node:pull token for registry-1.docker.io
=> [app internal] load .dockerignore
=> => transferring context: 2B
=> [app 1/5] FROM docker.io/library/node:18@sha256:ca07c02d13baf021ff5aadb3b48bcd1fcdd454826266ac313ce858676e8c1548
=> [app internal] load build context
=> => transferring context: 3.05MB
=> CACHED [app 2/5] WORKDIR /usr/src/app
=> CACHED [app 3/5] COPY package*.json ./
=> CACHED [app 4/5] RUN npm install
=> [app 5/5] COPY . .
=> [app] exporting to image
=> => exporting layers
=> => writing image sha256:ee8d780aca6212bdbc4fd02d8640232aec2b93a7e4b410843466cdc3ea52c5cf
=> => naming to docker.io/library/desafio10-app
[+] Running 2/2
✔ Container mongodb Started
✔ Container desafio10-app Started
```

- **--build:** Indica a Docker Compose que debe construir las imágenes antes de levantar los contenedores.
  - **-d o --detach:** Ejecuta los contenedores en segundo plano, liberando la terminal para otros comandos.
- **Verificación del Estado de los Contenedores**












Para verificar el estado de tus contenedores después de haber ejecutado el comando anterior usamos el comando:

“docker ps”

```
(base) wilsonorlandoquesadamoncayo@MAC-WQUESAD Desafio10 % docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED NAMES	STATUS	PORTS
f296e5177138	desafio10-app	"docker-entrypoint.s..."	7 minutes ago desafio10-app	Up 7 minutes	0.0.0.0:3000->3000/tcp
55f2202342f9	mongo:latest	"docker-entrypoint.s..."	31 minutes ago mongo	Up 7 minutes	0.0.0.0:27017->27017/tcp

En Docker desktop:

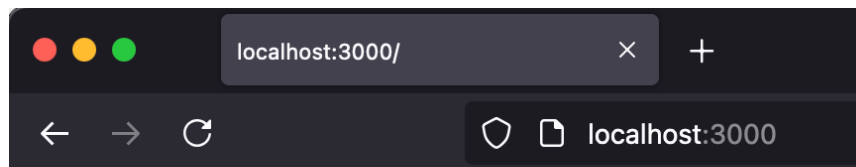
<input type="checkbox"/>	Name	Image	Status	Port(s)	CPU (%)	Actions
<input type="checkbox"/>	 <b>desafio10</b>		Running (2/2)		4.49%	  
<input type="checkbox"/>	 <b>mongodb</b>	mongo:late	Running	27017:27017	4.11%	  
<input type="checkbox"/>	 <b>app</b>	desafio10-i	Running	3000:3000	0.38%	  

- **Verifica Funcionamiento y Resultados exitosos**

Para probar la aplicación se puede en un navegador web o con una herramienta como curl o Postman para hacer una solicitud GET a http://localhost:3000.

Debería recibir una respuesta con el string 'Hello World!'.

```
(base) wilsonorlandoquesadamoncayo@MAC-WQUESAD ~ % curl http://localhost:3000
Hello World!%
(base) wilsonorlandoquesadamoncayo@MAC-WQUESAD ~ %
```



Hello World!