

Desafio #5

Objetivo:

El siguiente desafío tiene como objetivo desarrollar un build en docker y configurar un entorno local para que corra una aplicación con docker-compose.

Escenario:

Durante el sprint celebrado recientemente nuestro equipo nos asignó una tarea para desarrollar el archivo de build de una aplicación NestJS, esperan que para finalizar el sprint entreguemos el archivo Dockerfile funcional y un manifiesto de docker-compose que levante la aplicación y una base de datos MongoDB.

Nuestro aporte al equipo va a permitir que todos los desarrolladores que trabajen en el proyecto puedan levantar el mismo entorno para desarrollos locales.

La aplicación que va a ser manejada por este proceso se encuentra en el siguiente enlace:

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

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.
-

**Aclaración: este trabajo fue realizado desde WSL2 en Windows 10*

Paso 1: Clonar Repositorio

En este paso se realizó la clonación del repositorio suministrado para este trabajo.

Comando utilizado;

`$ git clone https://github.com/yosoyfunes/app-template-nestjs`

Paso 2: Creación de Dockerfile

Se genera el archivo Dockerfile y se lo setea de la siguiente manera

Dockerfile;

```
FROM node:18-alpine # Versión de NodeJS
```

```
WORKDIR /app # Directorio de trabajo
```

```
COPY /app-template-nestjs/* . # Copia todos los archivos del directorio al contenedor
```

```
RUN npm install # Instala dependencias
```

```
RUN npm run build # Construye aplicación
```

```
CMD ["npm", "run", "start"] # Ejecuta comandos para inicio de servidor
```

Comando utilizado para ejecutar el archivo Dockerfile;

**Estando ubicado en el directorio donde se encuentra nuestro archivo.*

\$ docker build .

```
Dockerfile X docker-compose.yml
Dockerfile > ...
1 FROM node:18-alpine
2
3 WORKDIR /app
4
5 COPY /app-template-nestjs/* .
6
7 RUN npm install
8
9 RUN npm run build
10
11 CMD ["npm", "run", "start"]
12

brian@DESKTOP-TJMT11B: ~/Desafio_5
brian@DESKTOP-TJMT11B:~/Desafio_5$ docker build .
[+] Building 1.9s (10/10) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 169B                             0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine 1.3s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [1/5] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e27 0.0s
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e27 0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 3.90kB                                0.0s
=> CACHED [2/5] WORKDIR /app                                    0.0s
=> CACHED [3/5] COPY /app-template-nestjs/* .                  0.0s
=> CACHED [4/5] RUN npm install                                 0.0s
=> CACHED [5/5] RUN npm run build                              0.0s
=> exporting to image                                           0.2s
=> => exporting layers                                           0.0s
=> => exporting manifest sha256:27b39d4ef9f38f96f457271339db093574809bafc594aec7975c4a19df9d2c51 0.0s
=> => exporting config sha256:167df0d5e1f09325d34651fbc77adfe27da1e6bcae01fb244aa8b60faa69188 0.0s
=> => exporting attestation manifest sha256:f4c4da665b873b667abb589f7c9dac0b43772d1c989355b6653f92e7acb94af 0.1s
=> => exporting manifest list sha256:aac5b20fb095f61d6a98e8c12bb9865795c897cc99790144425c75d91b9f5492 0.0s
=> => naming to moby-dangling@sha256:aac5b20fb095f61d6a98e8c12bb9865795c897cc99790144425c75d91b9f5492 0.0s
=> => unpacking to moby-dangling@sha256:aac5b20fb095f61d6a98e8c12bb9865795c897cc99790144425c75d91b9f5492 0.0s
brian@DESKTOP-TJMT11B:~/Desafio_5$
```

Paso 3: Creación Docker-Compose

Se crea el archivo `docker-compose.yml` con la siguiente estructura;

services:

mongodb:

image: mongo:latest # Utiliza la imagen más reciente de mongodb

container_name: mongodb # Nombra al contenedor

ports:

- "27017:27017" # Expone el puerto de mongodb en el equipo host

app:

build:

context: . # Utiliza el Dockerfile en la carpeta actual para construir
la imagen

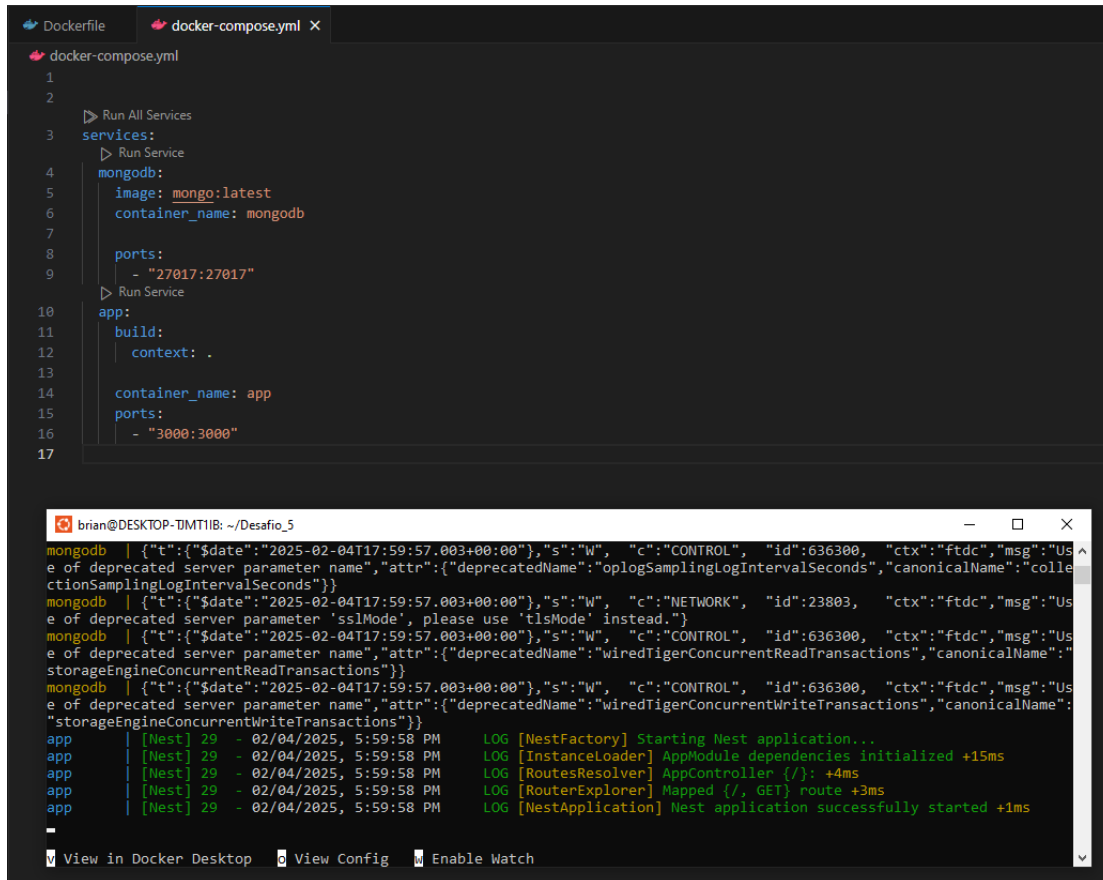
container_name: app # Nombra al contenedor

ports:

- "3000:3000" # Expone el puerto 3000 del equipo host

Comando de ejecución;

`$ docker-compose up`



The screenshot shows a code editor with a `docker-compose.yml` file and a terminal window displaying the logs of the services started by `docker-compose up`.

docker-compose.yml

```
1
2
3 services:
4   > Run Service
5   mongodb:
6     image: mongo:latest
7     container_name: mongodb
8
9     ports:
10      - "27017:27017"
11
12   > Run Service
13   app:
14     build:
15       context: .
16
17     container_name: app
18     ports:
19      - "3000:3000"
```

Terminal Output (brian@DESKTOP-TJMT1IB: ~/Desafio_5)

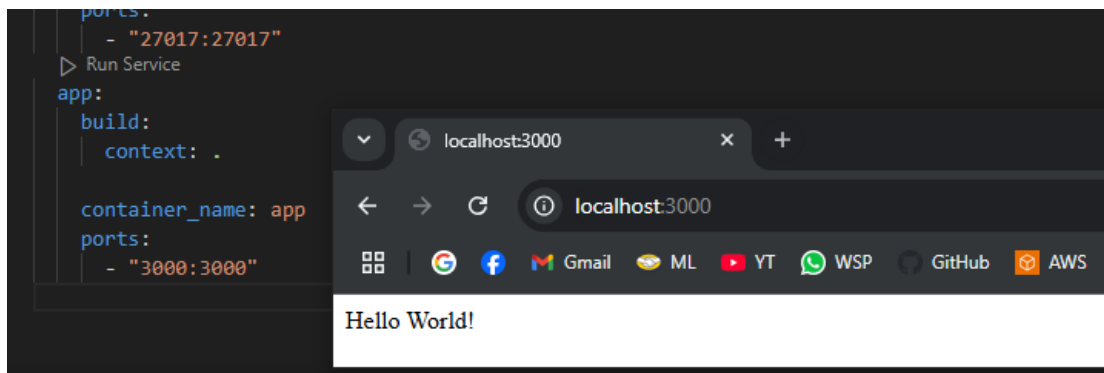
```
mongodb | {"t":{"$date":"2025-02-04T17:59:57.003+00:00"},"s":"W", "c":"CONTROL", "id":636300, "ctx":"ftdc","msg":"Use of deprecated server parameter name","attr":{"deprecatedName":"oplogSamplingLogIntervalSeconds","canonicalName":"collectionSamplingLogIntervalSeconds"}}
mongodb | {"t":{"$date":"2025-02-04T17:59:57.003+00:00"},"s":"W", "c":"NETWORK", "id":23803, "ctx":"ftdc","msg":"Use of deprecated server parameter 'sslMode', please use 'tlsMode' instead."}
mongodb | {"t":{"$date":"2025-02-04T17:59:57.003+00:00"},"s":"W", "c":"CONTROL", "id":636300, "ctx":"ftdc","msg":"Use of deprecated server parameter name","attr":{"deprecatedName":"wiredTigerConcurrentReadTransactions","canonicalName":"storageEngineConcurrentReadTransactions"}}
mongodb | {"t":{"$date":"2025-02-04T17:59:57.003+00:00"},"s":"W", "c":"CONTROL", "id":636300, "ctx":"ftdc","msg":"Use of deprecated server parameter name","attr":{"deprecatedName":"wiredTigerConcurrentWriteTransactions","canonicalName":"storageEngineConcurrentWriteTransactions"}}
app | [Nest] 29 - 02/04/2025, 5:59:58 PM LOG [NestFactory] Starting Nest application...
app | [Nest] 29 - 02/04/2025, 5:59:58 PM LOG [InstanceLoader] AppModule dependencies initialized +15ms
app | [Nest] 29 - 02/04/2025, 5:59:58 PM LOG [RoutesResolver] ApplicationController {}: +4ms
app | [Nest] 29 - 02/04/2025, 5:59:58 PM LOG [RouterExplorer] Mapped {/, GET} route +3ms
app | [Nest] 29 - 02/04/2025, 5:59:58 PM LOG [NestApplication] Nest application successfully started +1ms
```

At the bottom of the terminal, there are links: `View in Docker Desktop`, `View Config`, and `Enable Watch`.

Paso 4: Test de funcionamiento

Se verifica el correcto funcionamiento de los archivos creados;

Se puede observar por el puerto 3000 en "localhost:3000" que la app de nestjs está activa



brian@DESKTOP-TJMT1IB: ~/Desafio_5

```
sions":2},"execution":{"admissions":3},"workingMillis":125,"durationMillis":125}}
ngodb | {"t":{"$date":"2025-02-04T17:59:57.003+00:00"},"s":"W", "c":"CONTROL", "id":636
of deprecated server parameter name","attr":{"deprecatedName":"internalQueryCacheSize","ca
heMaxEntriesPerCollection"}}
```

En cuanto a la DB, se ejecuta el siguiente comando para realizar una conexión local;

`$ mongosh "mongodb://localhost:27017"`

