

Desafio # 5

Fecha de entrega: 18/11/2024

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 entregemos 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.

Entregables:

Los entregables establecidos para este proyecto con:

1. Código fuente de todo lo producido.

- a. Pueden hacer un fork del repositorio y subir los archivos necesarios
2. Documentacion.
3. Evidencia de las pruebas con resultado exitoso.

Evaluacion:

- Entrega en fecha.
- Redactar documentación legible y que sea comprendida por terceros..
- Añade material de soporte adicional.
 - Ejemplo: Diagrama de alto nivel.
- Cumple con las consignas solicitadas.
- El entregable es funcional.
 - Ejemplo: el archivo docker-compose al ejecutarse funciona sin errores y realiza lo solicitado.

Documentos de referencia:

- [Documentación NestJS.](#)
- [Documentacion Docker Compose.](#)
- [Mongo docker-hub](#)

Resolución:

1er paso: Clonar el codigo localmente desde

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

1er seccion: App

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

Dockerfile:

```
FROM node:18-alpine # aunque en un escenario real habría que  
confirmar la versión de node.js compatible
```

```
WORKDIR /app # todas las operaciones que se realicen después de  
esta instrucción se ejecutarán dentro de /app
```

```
COPY /app-template-nestjs/* . # Copiamos todo el código de la app  
al contenedor.
```

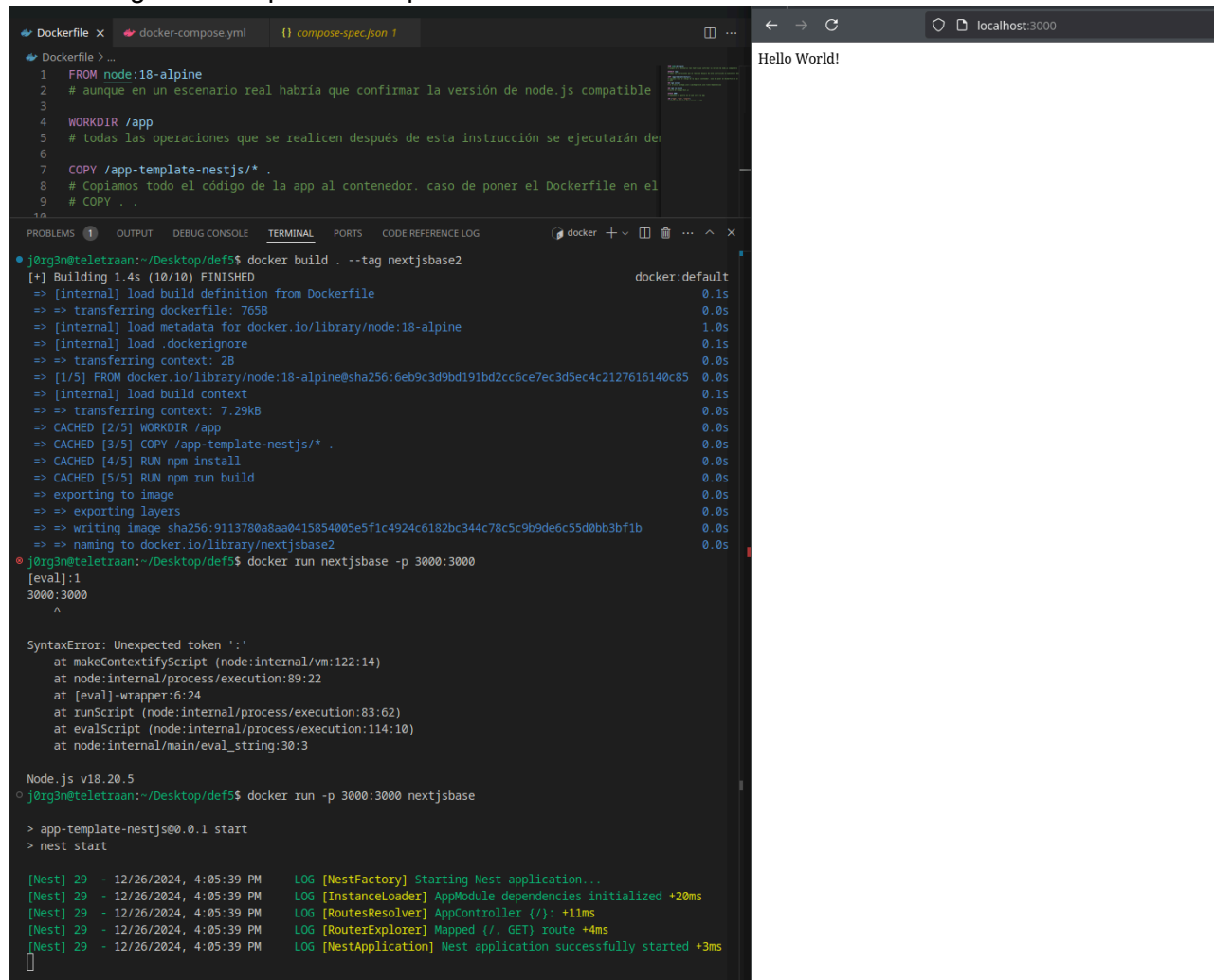
```
RUN npm install #el archivo package.json y package-lock.json  
tiene dependencias
```

RUN npm run build # build de la app Next.js

EXPOSE 3000 # Exponemos el puerto en el que corre la app

CMD ["npm", "run", "start"] # Comando por defecto para iniciar la app

Prueba: sigue una captura de la prueba del Dockerfile



The screenshot shows a terminal window with the following commands and output:

```
j0rg3n@teletraan:~/Desktop/def$ docker build . --tag nextjsbase2
[+] Building 1.4s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 765B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/node:18-alpine@sha256:6eb9c3d9bd191bd2cc6ce7ec3d5ec4c2127616140c85
=> [internal] load build context
=> => transferring context: 7.29kB
=> CACHED [2/5] WORKDIR /app
=> CACHED [3/5] COPY /app-template-nestjs/* .
=> CACHED [4/5] RUN npm install
=> CACHED [5/5] RUN npm run build
=> exporting to image
=> => exporting layers
=> => writing image sha256:9113780a8aa0415854005e5f1c4924c6182bc344c78c5c9b9de6c55d0bb3bf1b
=> => naming to docker.io/library/nextjsbase2

j0rg3n@teletraan:~/Desktop/def$ docker run nextjsbase -p 3000:3000
[eval]:1
3000:3000
^
SyntaxError: Unexpected token ':'
    at makeContextifyScript (node:internal/vm:122:14)
    at node:internal/process/execution:89:22
    at [eval]-wrapper:6:24
    at runScript (node:internal/process/execution:83:62)
    at evalScript (node:internal/process/execution:114:10)
    at node:internal/main/eval_string:30:3

Node.js v18.20.5
j0rg3n@teletraan:~/Desktop/def$ docker run -p 3000:3000 nextjsbase
> app-template-nestjs@0.0.1 start
> nest start

[Nest] 29 - 12/26/2024, 4:05:39 PM LOG [NestFactory] Starting Nest application...
[Nest] 29 - 12/26/2024, 4:05:39 PM LOG [InstanceLoader] AppModule dependencies initialized +20ms
[Nest] 29 - 12/26/2024, 4:05:39 PM LOG [RoutesResolver] AppController {}: +11ms
[Nest] 29 - 12/26/2024, 4:05:39 PM LOG [RouterExplorer] Mapped {/, GET} route +4ms
[Nest] 29 - 12/26/2024, 4:05:39 PM LOG [NestApplication] Nest application successfully started +3ms
```

To the right of the terminal, a web browser window is open to localhost:3000, displaying "Hello World!"

Docker Compose:

v1:

```
services:
  app:
    build:
      context: . # intentará buildear el Dockerfile en el mismo
      directorio
    container_name: app
    ports:
      - "3000:3000"
```

Prueba: sigue una captura de la prueba del compose v1

The screenshot shows a VS Code editor with three tabs: Dockerfile, docker-compose.yml, and compose-spec.json. The docker-compose.yml file is open, showing a service named 'app' with a build context of '.', container name 'app', and port '3000:3000'. The terminal at the bottom shows the command 'docker-compose up' being executed. The output indicates that the image for 'app' was built from the Dockerfile. The terminal also shows the application starting successfully with logs from NestJS.

```
services:
  app:
    build:
      context: .
      container_name: app
    ports:
      - "3000:3000"
```

```
j0rg3n@teletraan:~/Desktop/def5$ docker-compose up
WARNING: Found orphan containers (edb150d138aa_nextjs-app) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up.
Building app
[+] Building 2.0s (10/10) FINISHED
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 765B 0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine 1.5s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/5] FROM docker.io/library/node:18-alpine@sha256:6eb9c3d9bd191bd2cc6ce7ec3d5ec4c2127616140c85 0.0s
=> [internal] load build context 0.1s
=> => transferring context: 7.29kB 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 0.0s
=> => exporting layers 0.0s
=> writing image sha256:9113780a8aa0415854005e5f1c4924c6182bc344c78c5c9b9de6c55d0bb3bf1b 0.0s
=> => naming to docker.io/library/def5_app 0.0s
WARNING: Image for service app was built because it did not already exist. To rebuild this image you must use 'docker-compose build' or 'docker-compose up --build'.
Creating app ... done
Attaching to app
app |
app | > app-template-nestjs@0.0.1 start
app | > nest start
app |
app | [Nest] 29 - 12/26/2024, 4:14:26 PM LOG [NestFactory] Starting Nest application...
app | [Nest] 29 - 12/26/2024, 4:14:26 PM LOG [InstanceLoader] AppModule dependencies initialized +22ms
app | [Nest] 29 - 12/26/2024, 4:14:26 PM LOG [RoutesResolver] ApplicationController {}: +6ms
app | [Nest] 29 - 12/26/2024, 4:14:26 PM LOG [RouterExplorer] Mapped {/, GET} route +5ms
app | [Nest] 29 - 12/26/2024, 4:14:26 PM LOG [NestApplication] Nest application successfully start
app |
ed +3ms
```

2da sección: DB

Dado que no se proveen requerimientos de autenticación o variables de entorno para comunicar la app con la bd, se puede hacer:

v2:

```
services:
  mongodb:
    image: mongo:latest
    container_name: mongodb
```

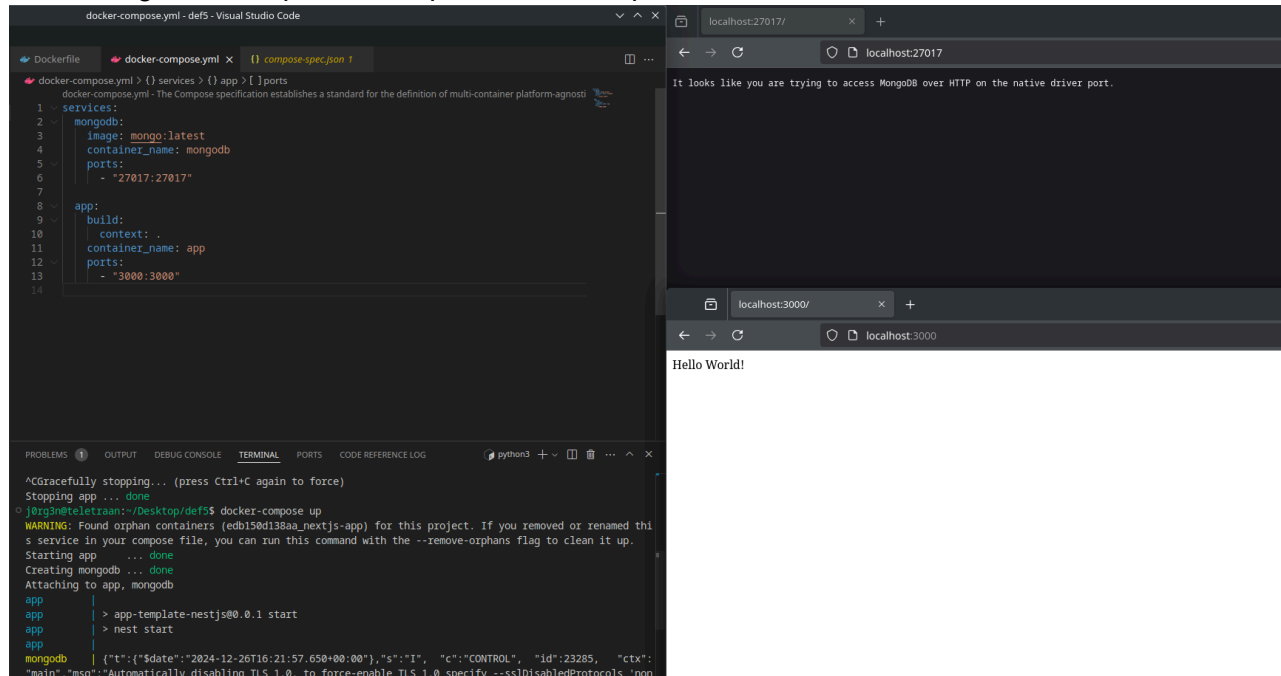
```

ports:
- "27017:27017"

app:
build:
  context: .
  container_name: app
ports:
- "3000:3000"

```

Prueba: sigue una captura de la prueba del compose v2



Prueba: sigue una captura de la prueba de conexión desde el container app al de mongodb

