

## Desafío # 10

**Realizado por:** Joselin Teixeira

*Fecha de entrega:* 27/08/2024

### 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.

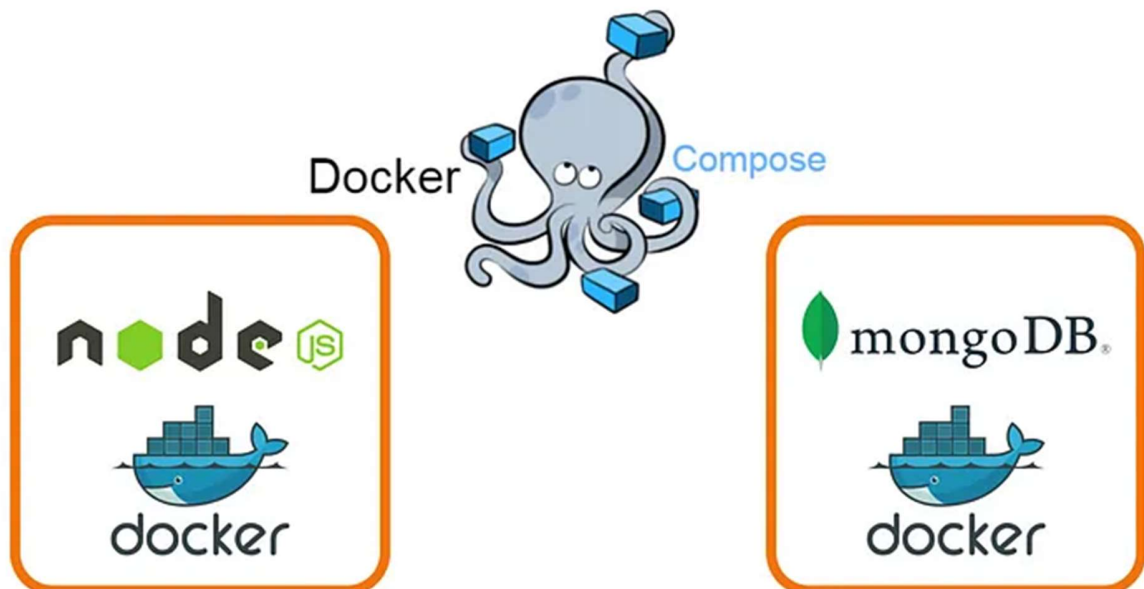
Nuestro aporte al equipo va a permitir que todos los desarrolladores que trabajen en el proyecto puedan poder levantar el mismo entorno en su entorno local.

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

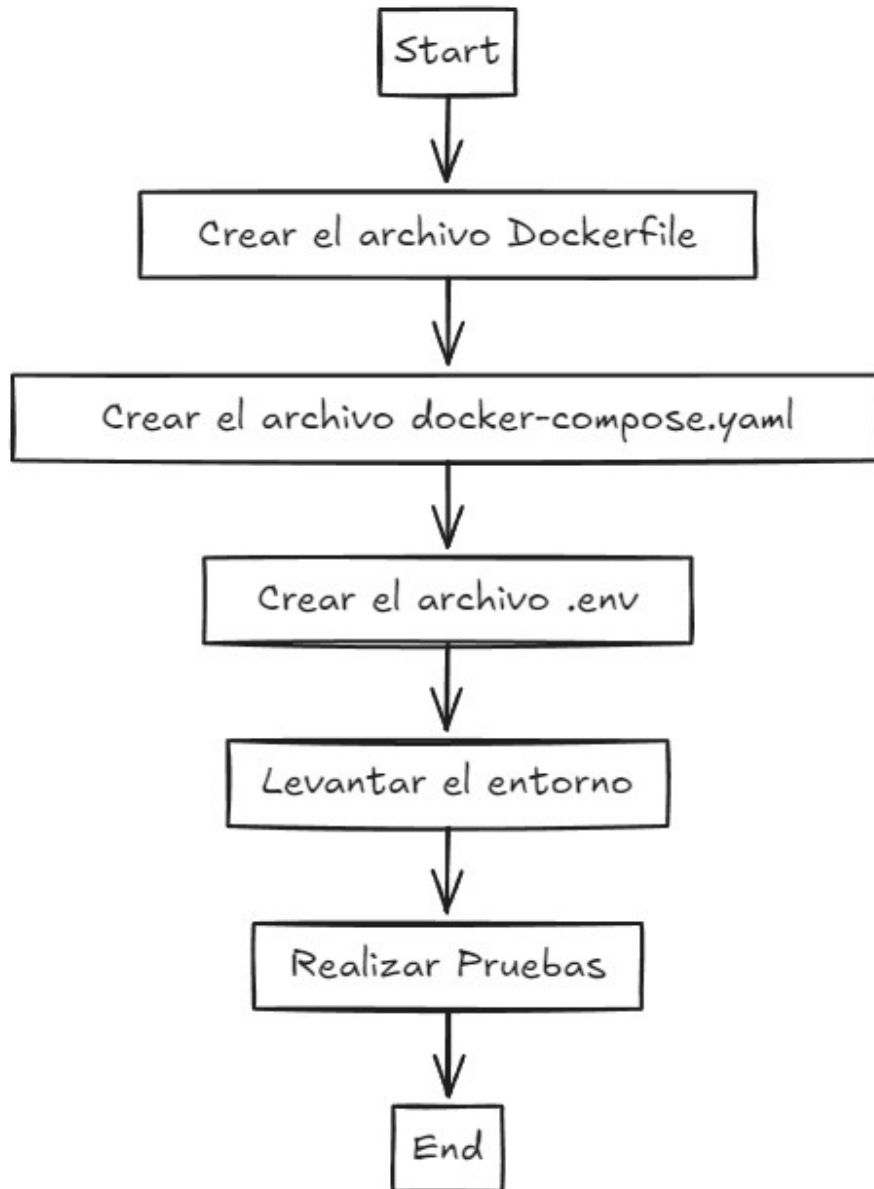
<https://github.com/edgaregonzalez/devops-bootcamp/tree/main/Desafios/Fase3/educacionit-app>

### 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.



## Diagrama de flujo



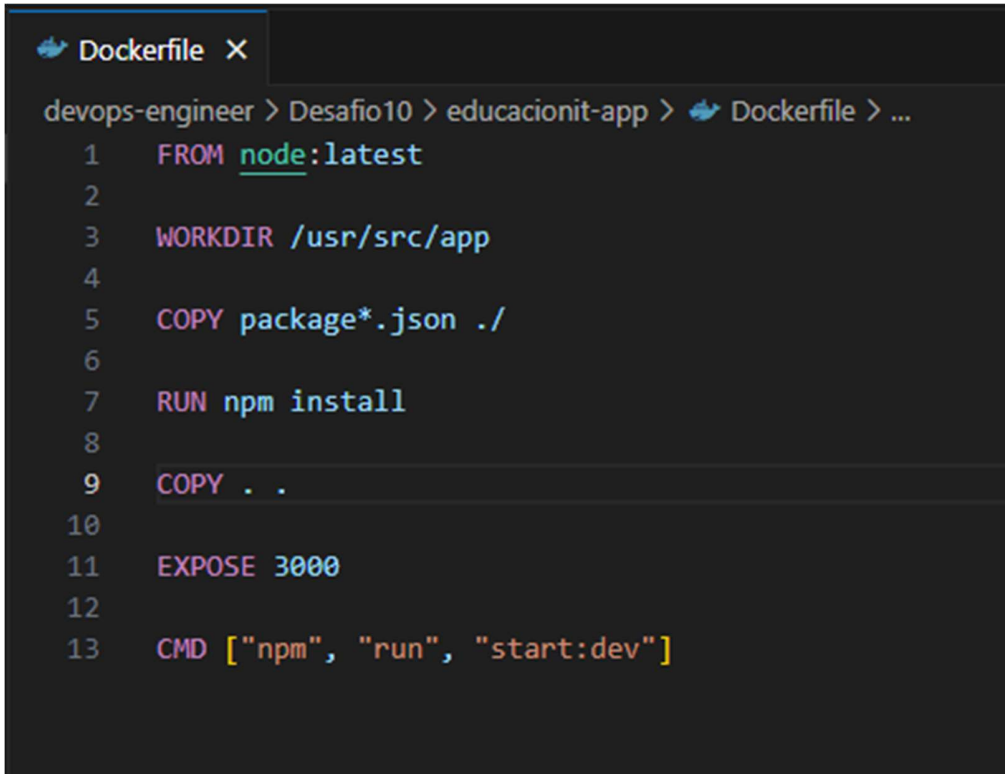
Para desarrollar un entorno local utilizando Docker y Docker Compose para una aplicación NestJS, seguiremos estos pasos:

- Crear el archivo Dockerfile.
- Crear el archivo docker-compose.yaml.
- Crear el archivo .env.
- Levantar el entorno. Ejecutar el builder
- Realizar pruebas y documentar el proceso.

A continuación, se detallan cada uno de estos pasos.

## Crear el archivo Dockerfile

El Dockerfile es un archivo de texto que contiene todas las instrucciones necesarias para construir la imagen de Docker de tu aplicación.



```
Dockerfile X
devops-engineer > Desafio10 > educacionit-app > Dockerfile > ...
1 FROM node:latest
2
3 WORKDIR /usr/src/app
4
5 COPY package*.json ./
6
7 RUN npm install
8
9 COPY . .
10
11 EXPOSE 3000
12
13 CMD ["npm", "run", "start:dev"]
```

# Usar una imagen base de Node.js – La Última Versión

**FROM node:latest**

Esta línea indica que la imagen base para construir tu contenedor será la imagen oficial de Node.js, específicamente la última versión.

# Directorio de trabajo

**WORKDIR /usr/src/app**

Esta instrucción establece el directorio de trabajo dentro del contenedor. Todas las siguientes instrucciones se ejecutarán en este directorio.

# Copiar los archivos de package.json y package-lock.json

**COPY package\*.json ./**

Esta línea copia los archivos package.json y package-lock.json desde nuestra máquina local al directorio de trabajo del contenedor. Estos archivos son esenciales para instalar las dependencias de la aplicación Node.js. Al copiarlos primero, optimizamos el proceso de construcción, ya que Docker puede utilizar la caché si no ha habido cambios en las dependencias.

```
# Instalar las dependencias  
RUN npm install
```

Esta instrucción ejecuta el comando `npm install` dentro del contenedor. Instalamos todas las dependencias definidas en `package.json`. Al hacerlo en esta etapa, nos aseguramos de que todas las bibliotecas estén disponibles para la aplicación.

```
# Copiar todos los archivos  
COPY . .
```

Esta línea copia todos los archivos y directorios desde nuestra máquina local al directorio de trabajo del contenedor.

```
# Exponer el puerto  
EXPOSE 3000
```

Esta instrucción indica que el contenedor escuchará en el puerto 3000. Facilita la comunicación entre el contenedor y el host (nuestra máquina local o cualquier otra). Al exponer el puerto, podremos acceder a la aplicación a través de `http://localhost:3000` en nuestro navegador.

```
# Comando para iniciar la aplicación  
CMD ["npm", "run", "start:dev"]
```

Esta línea especifica el comando que se ejecutará cuando se inicie el contenedor. En este caso, se está utilizando `npm run start:dev`, que generalmente inicia la aplicación en modo de desarrollo. Esto significa que la aplicación se reiniciará automáticamente si detecta cambios en el código.

## Crear el archivo `docker-compose.yml`.

El archivo `docker-compose.yml` es el que permite especificar y ejecutar múltiples contenedores de Docker definiendo sus servicios, redes y volúmenes. En este caso tendremos 2 servicios, uno para la aplicación NestJS y otro para MongoDB.

Crearemos 2 servicios:

**app** : Ejecuta el contenedor de la imagen `app` creada

**mongo** : Ejecuta el contenedor de la imagen `dbMongo`

```
docker-compose.yml X
devops-engineer > Desafio10 > educacionit-app > docker-compose.yml > ...
docker-compose.yml - The Compose specification establishes a standard for the definition of multi-container platform-agnostic applications (compose-spec.json)
1  ## Crea un docker-compose.yml que contenga los siguientes servicios:
2  ## - Un servicio llamado "app" que ejecute el contenedor de la imagen "app" que creaste en el desafío anterior.
3  ## - Un servicio llamado "db" que ejecute el contenedor de la imagen "db" que creaste en el desafío anterior
4
5  ## version: '3.8'
6
7  services:
8    app:
9      build:
10       context: .
11       dockerfile: Dockerfile
12     ports:
13       - "3000:3000"
14     environment:
15       - MONGO_DB_URI=mongodb://mongo:27017
16       - MONGO_DB_NAME=app-desafio10
17       - MONGO_DB_USER=root
18       - MONGO_DB_PASS=example
19     depends_on:
20       - mongo
21     volumes:
22       - ./usr/src/app
23
24     mongo:
25       image: mongo:latest
26       ports:
27         - "27017:27017"
28       environment:
29         - MONGO_INITDB_ROOT_USERNAME=root
30         - MONGO_INITDB_ROOT_PASSWORD=example
31       volumes:
32         - mongo-data:/data/db
33
34 volumes:
35   mongo-data:
```

Nota: Podemos omitir la versión

- **Servicio de la aplicación**

app:

build: Especifica que el servicio se construirá a partir de un Dockerfile. El context indica el directorio base para la construcción, y el dockerfile especifica el nombre del archivo Dockerfile.

ports: Mapea el puerto 3000 del contenedor al puerto 3000 del host.

environment: Define las variables de entorno necesarias para la aplicación, como la conexión a MongoDB.

depends\_on: Indica que el servicio de la aplicación depende del servicio de MongoDB.

volumes: Monta el directorio actual del host (.) al directorio /usr/src/app dentro del contenedor. Esto permite que los cambios en el código se reflejen automáticamente en el contenedor.

- **Servicio de MongoDB**

mongo:

image: Especificamos que se utilizará la imagen oficial de MongoDB, en su versión más reciente.

ports: Mapeamos el puerto 27017 del contenedor al puerto 27017 del host.

environment: Definimos las variables de entorno necesarias para inicializar MongoDB, como el nombre de usuario y contraseña del usuario root.

volumes: Montamos un volumen llamado mongo-data al directorio /data/db dentro del contenedor. Esto permite persistir los datos de MongoDB incluso si el contenedor se detiene o se elimina.

- **Definición de volúmenes**

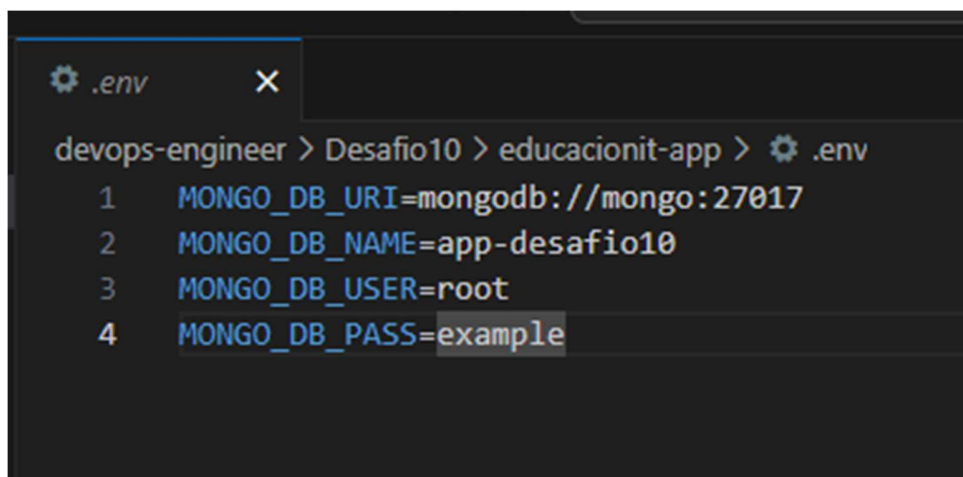
volumes:

mongo-data:

Definimos un volumen llamado mongo-data. Esto permitirá que los datos de MongoDB puedan quedarse fuera del ciclo de vida del contenedor. Los datos almacenados en este volumen se mantendrán incluso si el contenedor se detiene o se elimina.

## Crear el archivo .env

El archivo .env contendrá las variables de entorno que se utilizarán en la configuración de Docker Compose.

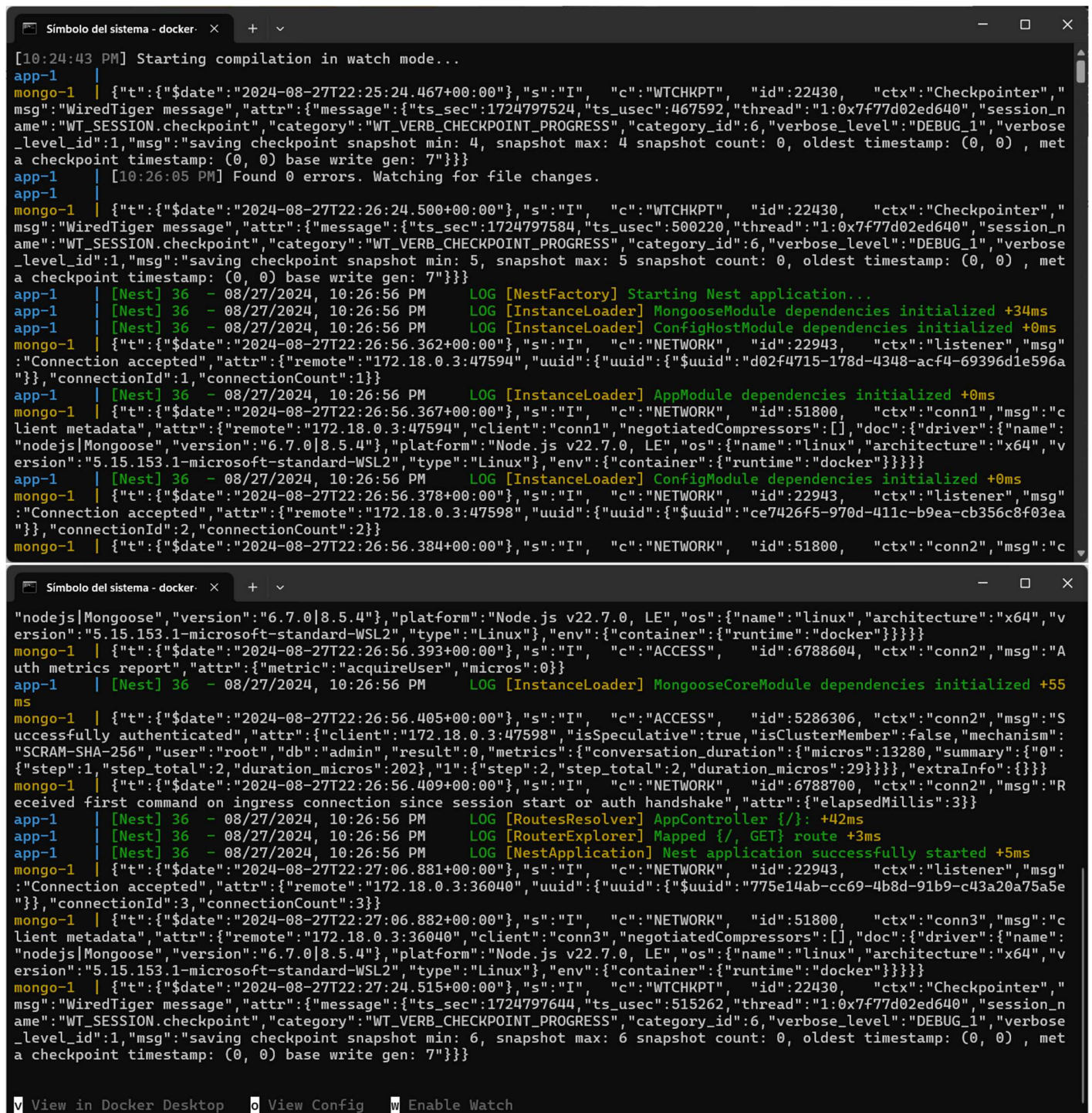


```
.env
devops-engineer > Desafio10 > educacionit-app > .env
1 MONGO_DB_URI=mongodb://mongo:27017
2 MONGO_DB_NAME=app-desafio10
3 MONGO_DB_USER=root
4 MONGO_DB_PASS=example
```



**Levantar el entorno:** Ejecutaremos el siguiente comando en la terminal para levantar la aplicación y la base de datos:

**docker-compose up -build**



```
[10:24:43 PM] Starting compilation in watch mode...
app-1 |
mongo-1 | {"t":{"$date":"2024-08-27T22:25:24.467+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpoint", "
msg":"WiredTiger message", "attr":{"message":{"ts_sec":1724797524,"ts_usec":467592,"thread":"1:0x7f77d02ed640","session_n
ame":"WT_SESSION.checkpoint", "category":"WT_VERB_CHECKPOINT_PROGRESS", "category_id":6, "verbose_level":"DEBUG_1", "verbose
_level_id":1, "msg":"saving checkpoint snapshot min: 4, snapshot max: 4 snapshot count: 0, oldest timestamp: (0, 0), met
a checkpoint timestamp: (0, 0) base write gen: 7"}}}
app-1 | [10:26:05 PM] Found 0 errors. Watching for file changes.
app-1 |
mongo-1 | {"t":{"$date":"2024-08-27T22:26:24.500+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpoint", "
msg":"WiredTiger message", "attr":{"message":{"ts_sec":1724797584,"ts_usec":500220,"thread":"1:0x7f77d02ed640","session_n
ame":"WT_SESSION.checkpoint", "category":"WT_VERB_CHECKPOINT_PROGRESS", "category_id":6, "verbose_level":"DEBUG_1", "verbose
_level_id":1, "msg":"saving checkpoint snapshot min: 5, snapshot max: 5 snapshot count: 0, oldest timestamp: (0, 0), met
a checkpoint timestamp: (0, 0) base write gen: 7"}}}
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM LOG [NestFactory] Starting Nest application...
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM LOG [InstanceLoader] MongooseModule dependencies initialized +34ms
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM LOG [InstanceLoader] ConfigHostModule dependencies initialized +0ms
mongo-1 | {"t":{"$date":"2024-08-27T22:26:56.362+00:00"},"s":"I", "c":"NETWORK", "id":22943, "ctx":"listener", "msg"
:"Connection accepted", "attr":{"remote":"172.18.0.3:47594", "uuid":{"$uuid":"d02f4715-178d-4348-acf4-69396d1e596a
"}}, "connectionId":1, "connectionCount":1}}
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM LOG [InstanceLoader] AppModule dependencies initialized +0ms
mongo-1 | {"t":{"$date":"2024-08-27T22:26:56.367+00:00"},"s":"I", "c":"NETWORK", "id":51800, "ctx":"conn1", "msg":"c
lient metadata", "attr":{"remote":"172.18.0.3:47594", "client":"conn1", "negotiatedCompressors":[], "doc":{"driver":{"name":
"nodejs|Mongoose", "version":"6.7.0|8.5.4"}, "platform":"Node.js v22.7.0, LE", "os":{"name":"linux", "architecture":"x64", "v
ersion":"5.15.153.1-microsoft-standard-WSL2", "type":"Linux"}, "env":{"container":{"runtime":"docker"}}}}}
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM LOG [InstanceLoader] ConfigModule dependencies initialized +0ms
mongo-1 | {"t":{"$date":"2024-08-27T22:26:56.378+00:00"},"s":"I", "c":"NETWORK", "id":22943, "ctx":"listener", "msg"
:"Connection accepted", "attr":{"remote":"172.18.0.3:47598", "uuid":{"$uuid":"ce7426f5-970d-411c-b9ea-cb356c8f03ea
"}}, "connectionId":2, "connectionCount":2}}
mongo-1 | {"t":{"$date":"2024-08-27T22:26:56.384+00:00"},"s":"I", "c":"NETWORK", "id":51800, "ctx":"conn2", "msg":"c
"nodejs|Mongoose", "version":"6.7.0|8.5.4"}, "platform":"Node.js v22.7.0, LE", "os":{"name":"linux", "architecture":"x64", "v
ersion":"5.15.153.1-microsoft-standard-WSL2", "type":"Linux"}, "env":{"container":{"runtime":"docker"}}}}}
mongo-1 | {"t":{"$date":"2024-08-27T22:26:56.393+00:00"},"s":"I", "c":"ACCESS", "id":6788604, "ctx":"conn2", "msg":"A
uth metrics report", "attr":{"metric":"acquireUser", "micros":0}}
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM LOG [InstanceLoader] MongooseCoreModule dependencies initialized +55
ms
mongo-1 | {"t":{"$date":"2024-08-27T22:26:56.405+00:00"},"s":"I", "c":"ACCESS", "id":5286306, "ctx":"conn2", "msg":"S
uccessfully authenticated", "attr":{"client":"172.18.0.3:47598", "isSpeculative":true, "isClusterMember":false, "mechanism":
"SCRAM-SHA-256", "user":"root", "db":"admin", "result":0, "metrics":{"conversation_duration":{"micros":13280, "summary":{"0":
{"step":1, "step_total":2, "duration_micros":202}, "1":{"step":2, "step_total":2, "duration_micros":29}}}}, "extraInfo":{}}}
mongo-1 | {"t":{"$date":"2024-08-27T22:26:56.409+00:00"},"s":"I", "c":"NETWORK", "id":6788700, "ctx":"conn2", "msg":"R
eceived first command on ingress connection since session start or auth handshake", "attr":{"elapsedMillis":3}}
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM LOG [RoutesResolver] AppController {/}: +42ms
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM LOG [RouterExplorer] Mapped {/, GET} route +3ms
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM LOG [NestApplication] Nest application successfully started +5ms
mongo-1 | {"t":{"$date":"2024-08-27T22:27:06.881+00:00"},"s":"I", "c":"NETWORK", "id":22943, "ctx":"listener", "msg"
:"Connection accepted", "attr":{"remote":"172.18.0.3:36040", "uuid":{"$uuid":"775e14ab-cc69-4b8d-91b9-c43a20a75a5e
"}}, "connectionId":3, "connectionCount":3}}
mongo-1 | {"t":{"$date":"2024-08-27T22:27:06.882+00:00"},"s":"I", "c":"NETWORK", "id":51800, "ctx":"conn3", "msg":"c
lient metadata", "attr":{"remote":"172.18.0.3:36040", "client":"conn3", "negotiatedCompressors":[], "doc":{"driver":{"name":
"nodejs|Mongoose", "version":"6.7.0|8.5.4"}, "platform":"Node.js v22.7.0, LE", "os":{"name":"linux", "architecture":"x64", "v
ersion":"5.15.153.1-microsoft-standard-WSL2", "type":"Linux"}, "env":{"container":{"runtime":"docker"}}}}}
mongo-1 | {"t":{"$date":"2024-08-27T22:27:24.515+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpoint", "
msg":"WiredTiger message", "attr":{"message":{"ts_sec":1724797644,"ts_usec":515262,"thread":"1:0x7f77d02ed640","session_n
ame":"WT_SESSION.checkpoint", "category":"WT_VERB_CHECKPOINT_PROGRESS", "category_id":6, "verbose_level":"DEBUG_1", "verbose
_level_id":1, "msg":"saving checkpoint snapshot min: 6, snapshot max: 6 snapshot count: 0, oldest timestamp: (0, 0), met
a checkpoint timestamp: (0, 0) base write gen: 7"}}}

View in Docker Desktop View Config Enable Watch
```

**docker compose ps**

Este comando muestra los contenedores

```
Simbolo del sistema x + v
C:\Users\josel\Documents\devops-engineer\Desafio10\educacionit-app>docker compose ps
NAME                                IMAGE                                COMMAND                                SERVICE    CREATED        STATUS        PORTS
educacionit-app-app-1              educacionit-app-app                "docker-entrypoint.s..."           app        2 hours ago    Up 2 hours    0.0.0.0:3000->3000/tcp
educacionit-app-mongo-1            mongo:latest                        "docker-entrypoint.s..."           mongo      2 hours ago    Up 2 hours    0.0.0.0:27017->27017/tcp

C:\Users\josel\Documents\devops-engineer\Desafio10\educacionit-app>
```

## docker-compose logs app

Comando para ver los registros de la aplicación.

```
Simbolo del sistema x + v
[10:24:43 PM] Starting compilation in watch mode...
app-1 |
app-1 | [10:26:05 PM] Found 0 errors. Watching for file changes.
app-1 |
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM      LOG [NestFactory] Starting Nest application...
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM      LOG [InstanceLoader] MongooseModule dependencies initialized +34ms
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM      LOG [InstanceLoader] ConfigHostModule dependencies initialized +0ms
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM      LOG [InstanceLoader] AppModule dependencies initialized +0ms
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM      LOG [InstanceLoader] ConfigModule dependencies initialized +0ms
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM      LOG [InstanceLoader] MongooseCoreModule dependencies initialized +55ms
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM      LOG [RoutesResolver] AppController {/}: +42ms
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM      LOG [RouterExplorer] Mapped {/, GET} route +3ms
app-1 | [Nest] 36 - 08/27/2024, 10:26:56 PM      LOG [NestApplication] Nest application successfully started +5ms

C:\Users\josel\Documents\devops-engineer\Desafio10\educacionit-app>
```

## docker-compose logs mongo

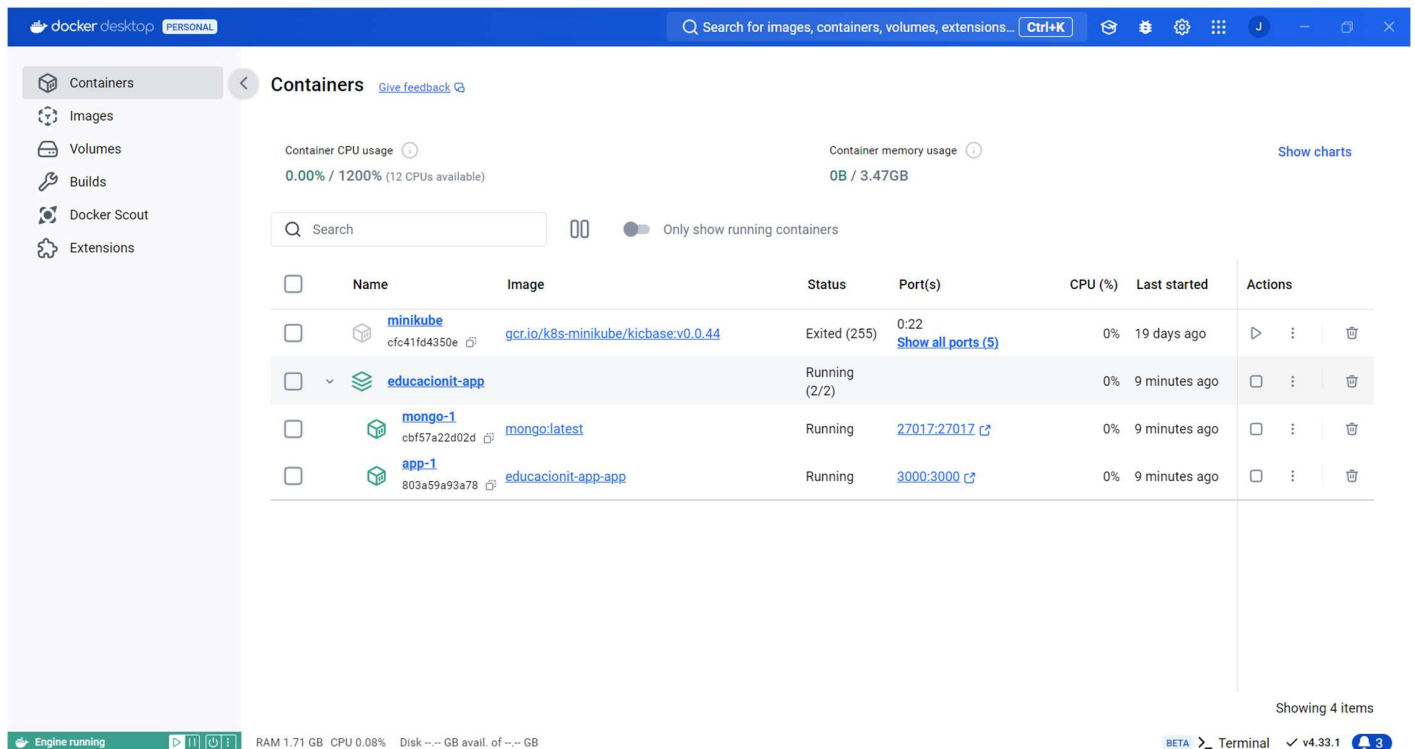
Para ver los registros de MongoDB.

```
Simbolo del sistema x + v
message":{"ts_sec":1724804546,"ts_usec":236278,"thread":"1:0x7f77d02ed640","session_name":"WT_SESSION.checkpoint","category":"WT_VERB_CHECKPOINT_PROGRESS","c
category_id":6,"verbose_level":"DEBUG_1","verbose_level_id":1,"msg":"saving checkpoint snapshot min: 121, snapshot max: 121 snapshot count: 0, oldest timesta
mp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 7}}}}
mongo-1 | {"t":{"$date":"2024-08-28T00:23:26.251+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpointner","msg":"WiredTiger message","attr":{"m
message":{"ts_sec":1724804606,"ts_usec":251307,"thread":"1:0x7f77d02ed640","session_name":"WT_SESSION.checkpoint","category":"WT_VERB_CHECKPOINT_PROGRESS","c
category_id":6,"verbose_level":"DEBUG_1","verbose_level_id":1,"msg":"saving checkpoint snapshot min: 122, snapshot max: 122 snapshot count: 0, oldest timesta
mp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 7}}}}
mongo-1 | {"t":{"$date":"2024-08-28T00:24:26.271+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpointner","msg":"WiredTiger message","attr":{"m
message":{"ts_sec":1724804666,"ts_usec":271468,"thread":"1:0x7f77d02ed640","session_name":"WT_SESSION.checkpoint","category":"WT_VERB_CHECKPOINT_PROGRESS","c
category_id":6,"verbose_level":"DEBUG_1","verbose_level_id":1,"msg":"saving checkpoint snapshot min: 123, snapshot max: 123 snapshot count: 0, oldest timesta
mp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 7}}}}
mongo-1 | {"t":{"$date":"2024-08-28T00:25:26.285+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpointner","msg":"WiredTiger message","attr":{"m
message":{"ts_sec":1724804726,"ts_usec":285769,"thread":"1:0x7f77d02ed640","session_name":"WT_SESSION.checkpoint","category":"WT_VERB_CHECKPOINT_PROGRESS","c
category_id":6,"verbose_level":"DEBUG_1","verbose_level_id":1,"msg":"saving checkpoint snapshot min: 124, snapshot max: 124 snapshot count: 0, oldest timesta
mp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 7}}}}
mongo-1 | {"t":{"$date":"2024-08-28T00:26:26.297+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpointner","msg":"WiredTiger message","attr":{"m
message":{"ts_sec":1724804786,"ts_usec":297605,"thread":"1:0x7f77d02ed640","session_name":"WT_SESSION.checkpoint","category":"WT_VERB_CHECKPOINT_PROGRESS","c
category_id":6,"verbose_level":"DEBUG_1","verbose_level_id":1,"msg":"saving checkpoint snapshot min: 125, snapshot max: 125 snapshot count: 0, oldest timesta
mp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 7}}}}
mongo-1 | {"t":{"$date":"2024-08-28T00:27:26.313+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpointner","msg":"WiredTiger message","attr":{"m
message":{"ts_sec":1724804846,"ts_usec":313504,"thread":"1:0x7f77d02ed640","session_name":"WT_SESSION.checkpoint","category":"WT_VERB_CHECKPOINT_PROGRESS","c
category_id":6,"verbose_level":"DEBUG_1","verbose_level_id":1,"msg":"saving checkpoint snapshot min: 126, snapshot max: 126 snapshot count: 0, oldest timesta
mp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 7}}}}
mongo-1 | {"t":{"$date":"2024-08-28T00:28:26.331+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpointner","msg":"WiredTiger message","attr":{"m
message":{"ts_sec":1724804906,"ts_usec":331265,"thread":"1:0x7f77d02ed640","session_name":"WT_SESSION.checkpoint","category":"WT_VERB_CHECKPOINT_PROGRESS","c
category_id":6,"verbose_level":"DEBUG_1","verbose_level_id":1,"msg":"saving checkpoint snapshot min: 127, snapshot max: 127 snapshot count: 0, oldest timesta
mp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 7}}}}
mongo-1 | {"t":{"$date":"2024-08-28T00:29:26.349+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpointner","msg":"WiredTiger message","attr":{"m
message":{"ts_sec":1724804966,"ts_usec":349653,"thread":"1:0x7f77d02ed640","session_name":"WT_SESSION.checkpoint","category":"WT_VERB_CHECKPOINT_PROGRESS","c
category_id":6,"verbose_level":"DEBUG_1","verbose_level_id":1,"msg":"saving checkpoint snapshot min: 128, snapshot max: 128 snapshot count: 0, oldest timesta
mp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 7}}}}
mongo-1 | {"t":{"$date":"2024-08-28T00:30:26.368+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpointner","msg":"WiredTiger message","attr":{"m
message":{"ts_sec":1724805026,"ts_usec":368425,"thread":"1:0x7f77d02ed640","session_name":"WT_SESSION.checkpoint","category":"WT_VERB_CHECKPOINT_PROGRESS","c
category_id":6,"verbose_level":"DEBUG_1","verbose_level_id":1,"msg":"saving checkpoint snapshot min: 129, snapshot max: 129 snapshot count: 0, oldest timesta
mp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 7}}}}
mongo-1 | {"t":{"$date":"2024-08-28T00:31:26.383+00:00"},"s":"I", "c":"WTCHKPT", "id":22430, "ctx":"Checkpointner","msg":"WiredTiger message","attr":{"m
message":{"ts_sec":1724805086,"ts_usec":383105,"thread":"1:0x7f77d02ed640","session_name":"WT_SESSION.checkpoint","category":"WT_VERB_CHECKPOINT_PROGRESS","c
category_id":6,"verbose_level":"DEBUG_1","verbose_level_id":1,"msg":"saving checkpoint snapshot min: 130, snapshot max: 130 snapshot count: 0, oldest timesta
mp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 7}}}}

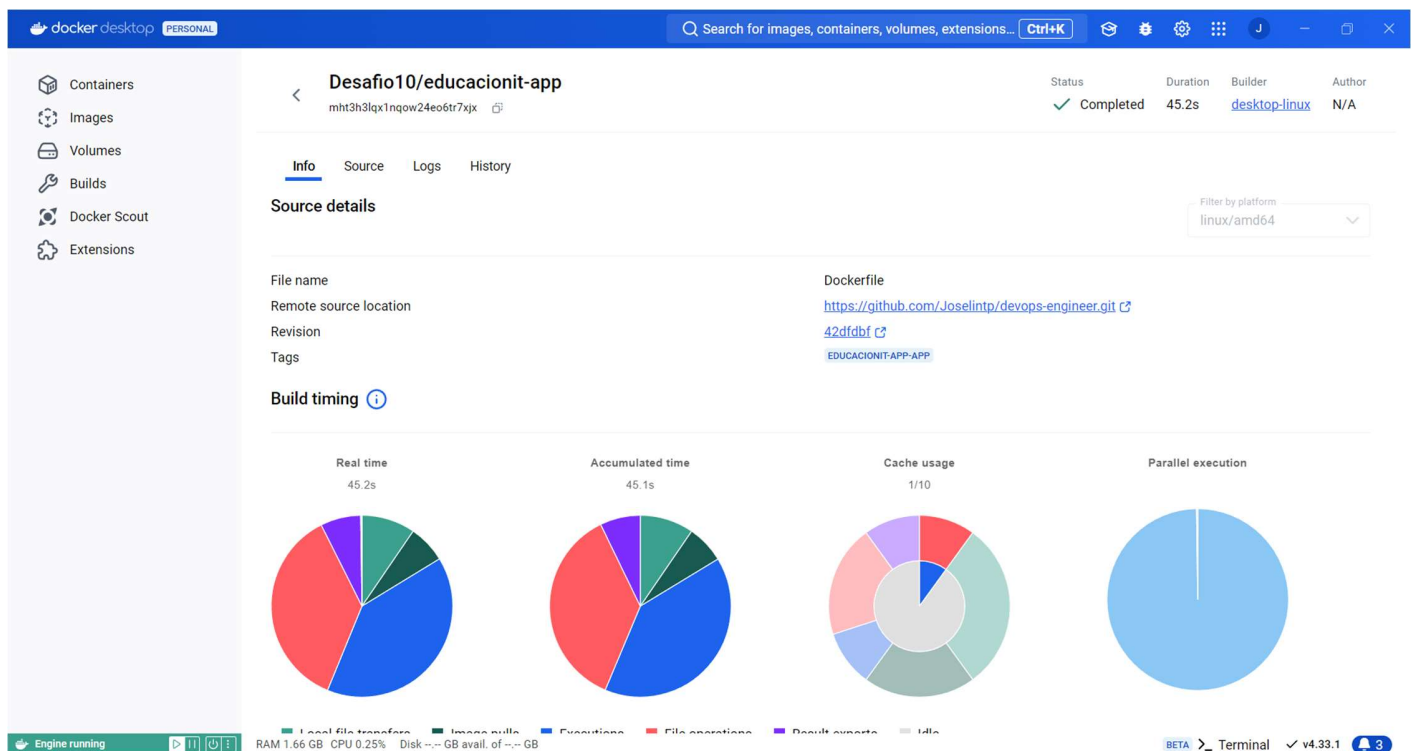
C:\Users\josel\Documents\devops-engineer\Desafio10\educacionit-app>
```

También desde Docker Desktop podremos consultar los contenedores creados:

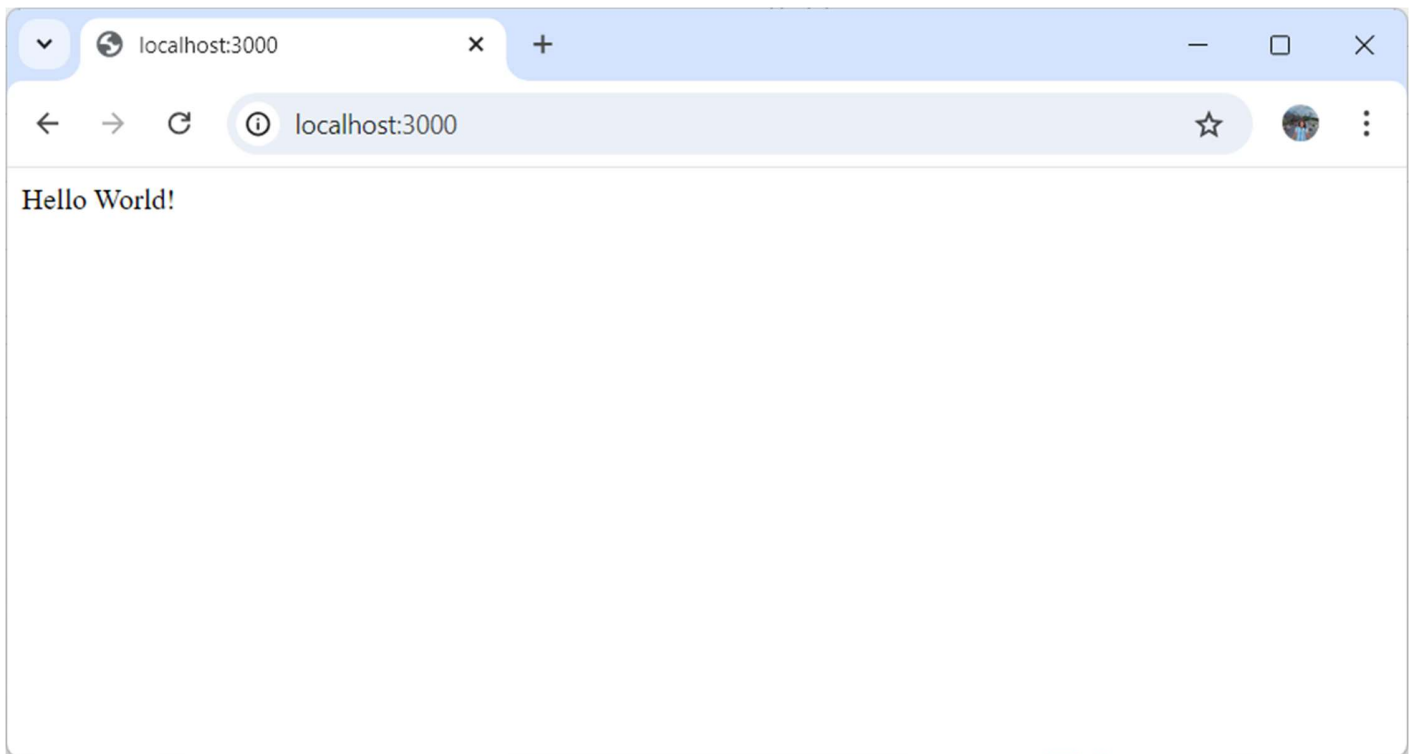




## Consultar los Builds



**Acceder a la aplicación:** Una vez que los contenedores estén en funcionamiento, podremos acceder a la aplicación en <http://localhost:3000>.



## Referencias:

[https://hub.docker.com/\\_/mongo](https://hub.docker.com/_/mongo)

<https://docs.docker.com/reference/dockerfile/>

<https://docs.docker.com/reference/cli/docker/compose/>