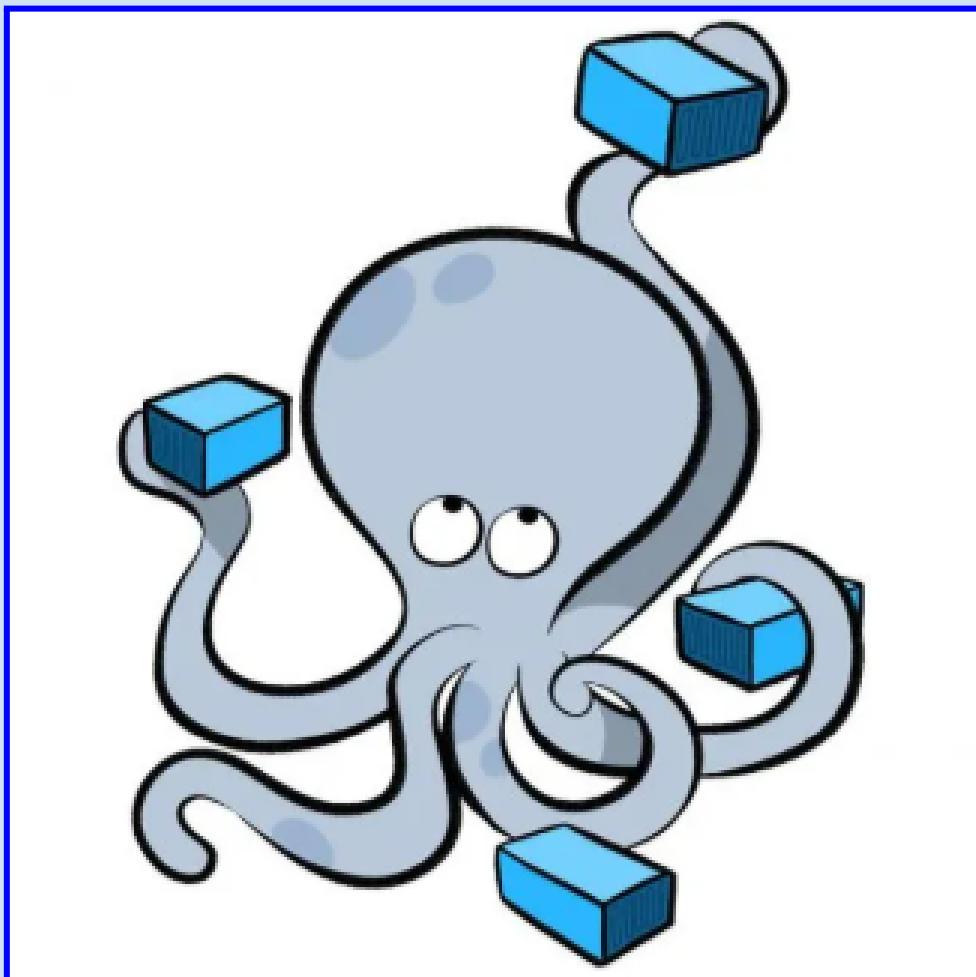


# Tarea Docker - Compose



# ÍNDICE

Caso práctico:.....	3
Enunciado:.....	3
1. Crea una nueva red que se llame “adaitsnetwork”.....	3
2. Crea un nuevo contenedor.....	4
3. Crear API REST en Node.js.....	5
4. Despliegue de los Servicios con Docker Compose.....	7
Indicaciones de entrega.....	8

## Caso práctico:

Eres un explorador digital en la misteriosa tierra de Docker-landia, donde los servicios viven en contenedores y las aplicaciones se despliegan con la velocidad del viento. Tu misión será construir una ciudad de servicios que interactúen armoniosamente entre sí.

En esta tarea en concreto deberás crear 2 servicios:

- Un servidor web NGINX que sirva una página web estática (web.html).
- Una API Rest en Node.js que responda a solicitudes GET.

## Enunciado:

### 1. Crea una nueva red que se llame “adaitsnetwork”.

Para crear una nueva red debemos abrir el cmd o Powershell y escribir el comando **<docker network create adaitsnetwork>**. Para comprobar que se haya creado la red correctamente ejecutamos **<docker network ls>** para ver nuestra lista de redes.

```
Microsoft Windows [Versión 10.0.26200.7019]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\vrida>docker network create adaitsnetwork
5d517fef70c0bce0c8b9cad21b64f571e8c28a836e65c2d3f7dc707344883c31

C:\Users\vrida>docker network ls
NETWORK ID      NAME                DRIVER      SCOPE
946fafffc1cad  Wordpress-trabajo-despliegue   bridge      local
5d517fef70c0    adaitsnetwork        bridge      local
26b24fac7b9a   bridge              bridge      local
e702edd42fb5   host                host       local
732e4010cd8d   none               null       local
8ca3698d77a6   redis              bridge      local

C:\Users\vrida>
```

## 2. Crea un nuevo contenedor.

Se deberá llamar “Servidor Web NGINX” (última versión) que por defecto inicie con una página html (web.html) que contendrá un botón que hará peticiones a una API con el contenido indicado en la imagen del anexo.

### 1. Crea una carpeta que contenga:

- web.html
- Dockerfile

Primero creamos una carpeta llamada “nginx-daw”, yo la creé en mi escritorio. Ahí creamos los archivos web.html y Dockerfile. Y nos aseguramos de que dentro de la carpeta aparecen los archivos.

```
Directorio de C:\Users\vrida\OneDrive\Escritorio\nginx-daw

06/11/2025 16:14    <DIR>      .
06/11/2025 16:13    <DIR>      ..
06/11/2025 16:16              65 Dockerfile
06/11/2025 16:16          580 web.html
                2 archivos       645 bytes
                2 dirs   247.701.041.152 bytes libres

C:\Users\vrida\OneDrive\Escritorio\nginx-daw>
```

### 2. Construir la imagen nginx-daw (docker build).

Usamos el cmd para construir la imagen con el comando [\*\*<docker built -t nginx-daw>\*\*](#).

```
C:\Users\vrida\OneDrive\Escritorio\nginx-daw>docker build -t nginx-daw .
[+] Building 2.4s (8/8) FINISHED                                            docker:desktop-linux
=> [internal] load build definition from Dockerfile                      0.0s
=> => transferring dockerfile: 112B                                         0.0s
=> [internal] load metadata for docker.io/library/nginx:latest           1.6s
=> [auth] library/nginx:pull token for registry-1.docker.io               0.0s
=> [internal] load .dockerrcignore                                         0.0s
=> => transferring context: 2B                                           0.0s
=> [internal] load build context                                         0.1s
=> => transferring context: 837B                                         0.1s
=> CACHED [1/2] FROM docker.io/library/nginx:latest@sha256:1beed3ca46acebe9d3fb62e9067f03d 0.0s
=> => resolve docker.io/library/nginx:latest@sha256:1beed3ca46acebe9d3fb62e9067f03d05d5bfa 0.0s
=> [2/2] COPY web.html /usr/share/nginx/html/web.html                     0.1s
=> => exporting to image                                                 0.4s
=> => exporting layers                                                 0.1s
=> => exporting manifest sha256:a6e0cb778c13c1845e0fa3cd967011ed6c793d5fe51142bc0f0093d7a4 0.0s
=> => exporting config sha256:8b9ae98529182a8ebfc52ca2c97828125a4ec24d548044d56eac3dc135b6 0.0s
=> => exporting attestation manifest sha256:f9c3de03f1c4778d2d7273e32ac67680f840ca51274553 0.0s
=> => exporting manifest list sha256:867d038ec55cb86998883062b807384d9a8d164d5f1c546660f35 0.0s
=> => naming to docker.io/library/nginx-daw:latest                         0.0s
=> => unpacking to docker.io/library/nginx-daw:latest                      0.1s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/6f8d2f75gcpt354ys
llptyrs2
```

### 3. Ejecutar contenedor con la imagen nginx-daw y en la red “adaitsnetwork”.

Para ejecutar el contenedor, usamos el comando `<docker run -d --name nginx-daw --network adaitsnetwork -p 80:80 nginx-daw>` que ya usamos antes.

Luego confirmamos que está en ejecución con el comando `<docker ps>`.

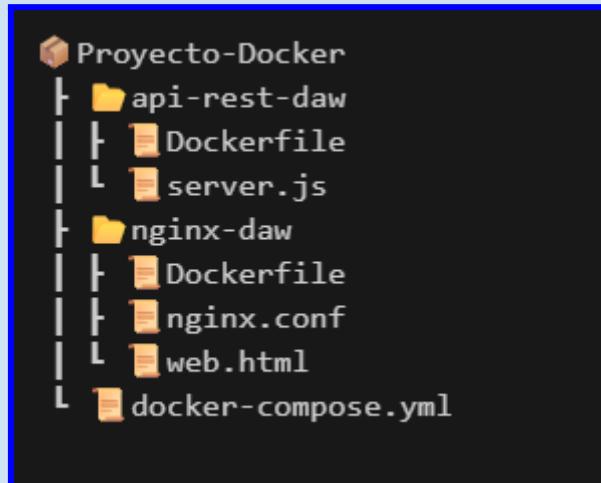
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
674c6a718217	proyecto-docker-api-rest	"docker-entrypoint.s..."	14 minutes ago	Up 14 minutes	0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp
3c08e8215a9e	proyecto-docker-web-nginx	"/docker-entrypoint..."	14 minutes ago	Up 14 minutes	0.0.0.0:8080->80/tcp, [::]:8080->80/tcp

### 3. Crear API REST en Node.js.

Se deberá crear una API REST en Node.js que contenga un server.js con el contenido indicado en la imagen del anexo.

#### 1. Crea una carpeta que contenga:

- server.js
- Dockerfile



Esta es la estructura del proyecto con las carpetas y los scripts necesarios.

## 2. Construir la imagen api-rest-daw (docker build)

Para ello ejecutamos el comando <docker build -t api-rest-daw ./api-rest-daw>

```
C:\Users\vrida\OneDrive\Escritorio\Proyecto-Docker>docker build -t api-rest-daw ./api-rest-daw
[+] Building 2.1s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 358B
=> [internal] load metadata for docker.io/library/node:latest
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/node:latest@sha256:0b1fad950f54a1d6f9e8e580205c157b43315d2c4231c3a0
=> => resolve docker.io/library/node:latest@sha256:0b1fad950f54a1d6f9e8e580205c157b43315d2c4231c3a0
=> [internal] load build context
=> => transferring context: 31B
=> CACHED [2/4] WORKDIR /app
=> CACHED [3/4] COPY server.js .
=> CACHED [4/4] RUN npm init -y && npm install express
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:3660c03e3eb67b7b60bb407052eae3d1e97bc63e34db2d390eb87a9d2604d638
```

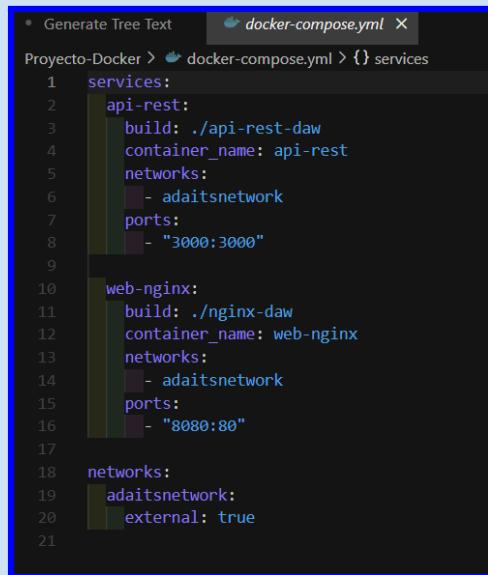
## 3. Ejecutar el contenedor con la imagen api-rest-daw y en la red “adaitstnetwork”.

Ejecutamos <docker-compose up -d> y para verlo en el navegador accedemos a <http://localhost:8080/web.html>



Para facilitar la creación de los contenedores, se debe realizar un docker-compose (.yml), el cual facilitará crear los servicios a la vez:

- NGINX [1.5 puntos]
- API [1.5 puntos]



```
* Generate Tree Text docker-compose.yml X
Proyecto-Docker > docker-compose.yml > {} services
1 services:
2   api-rest:
3     build: ./api-rest-daw
4     container_name: api-rest
5     networks:
6       - adaitsnetwork
7     ports:
8       - "3000:3000"
9
10  web-nginx:
11    build: ./nginx-daw
12    container_name: web-nginx
13    networks:
14      - adaitsnetwork
15    ports:
16      - "8080:80"
17
18 networks:
19   adaitsnetwork:
20     external: true
21
```

#### 4. Despliegue de los Servicios con Docker Compose

- Ejecuta “docker-compose up” para desplegar ambos servicios simultáneamente.

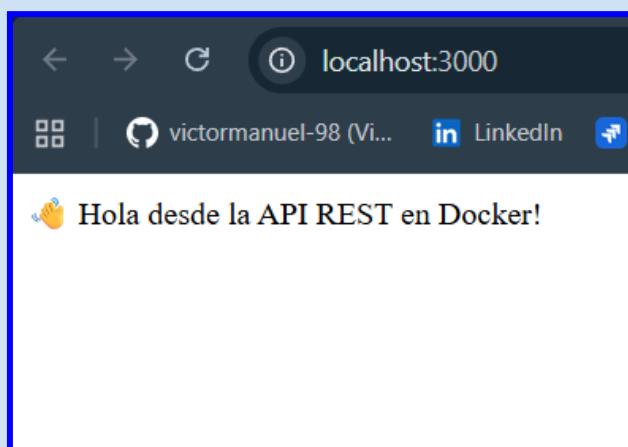
```
C:\Users\vrida\OneDrive\Escritorio\Proyecto-Docker>docker-compose up -d
[+] Running 2/2
✓ Container api-rest    Started
✓ Container web-nginx   Started

C:\Users\vrida\OneDrive\Escritorio\Proyecto-Docker>docker ps
```

- Verifica que el servidor web NGINX sirva correctamente la página estática al acceder a <http://localhost:8080/web.html>



- Verifica que la API REST responda a solicitudes GET en <http://localhost:3000>.



## Indicaciones de entrega

Deberás entregar un fichero '.pdf' con las capturas necesarias de cada apartado. Además, deberás añadir los archivos de configuración (\*.yml) que hayas tenido que realizar para el despliegue con docker-compose.