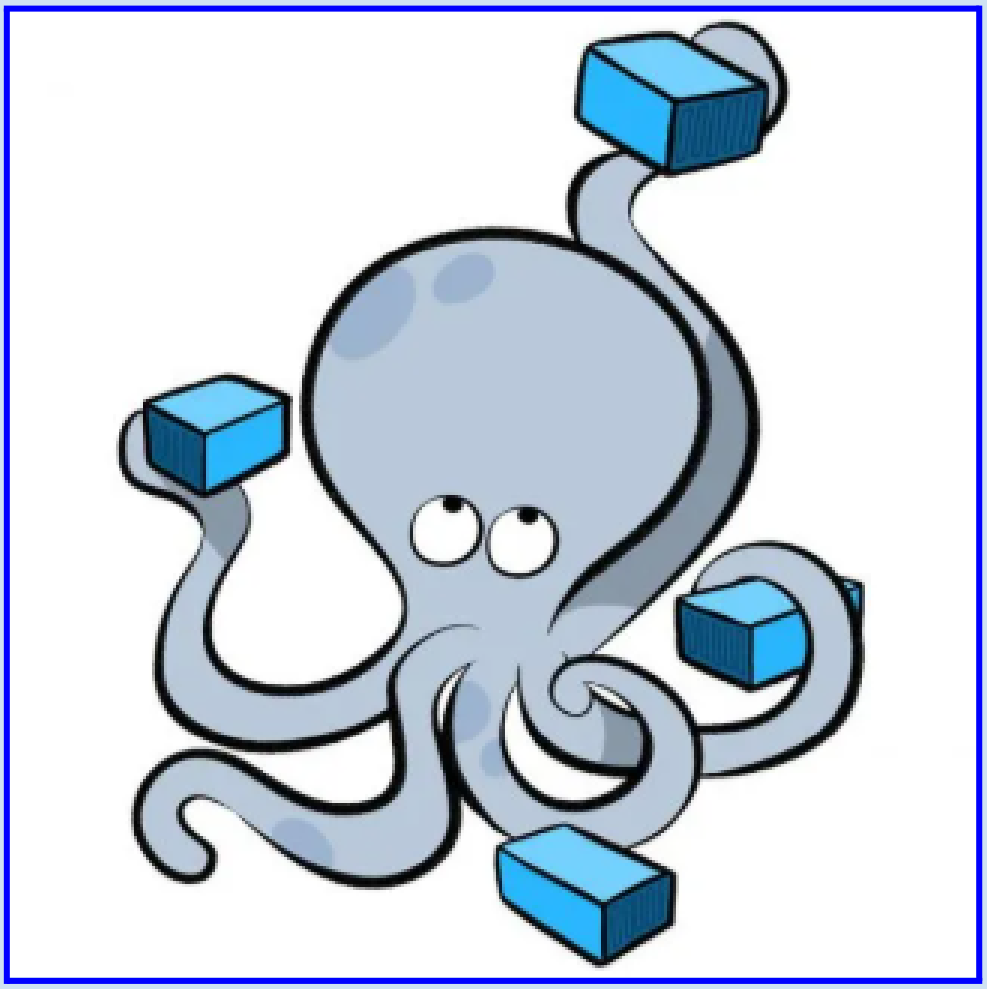
Víctor Manuel Ridao Chaves
28/10/2025
2º DAW ADAits

Tarea Docker - Compose



ÍNDICE

Caso práctico:

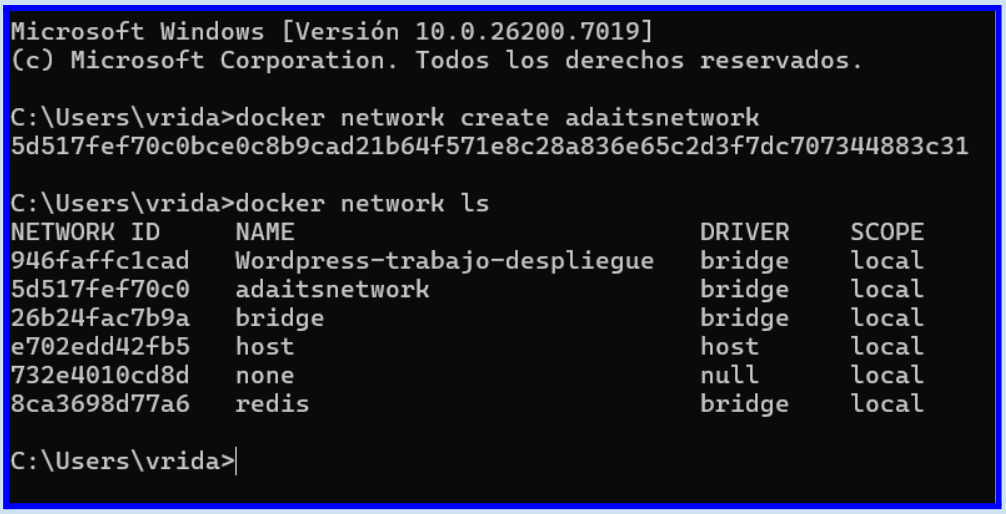
Eres un explorador digital en la misteriosa tierra de Docker-landia, donde los servicios viven en contenedores y las aplicaciones se despliegan con la velocidad del viento. Tu misión será construir una ciudad de servicios que interactúen armoniosamente entre sí.
En esta tarea en concreto deberás crear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



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.

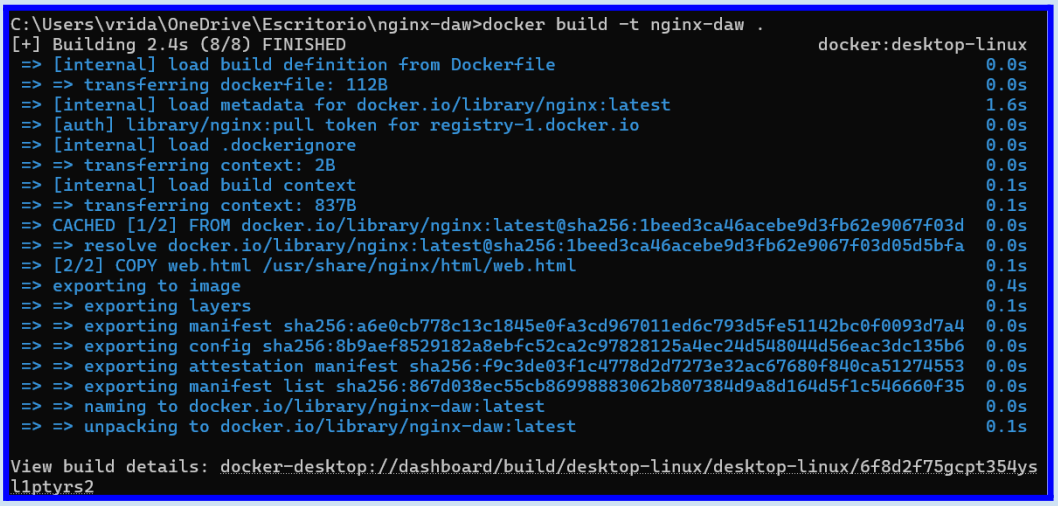
1. Crea una carpeta que contenga:

- **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.

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

Usamos el cmd para construir la imagen con el comando **<docker built -t nginx-daw>**.



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>**.

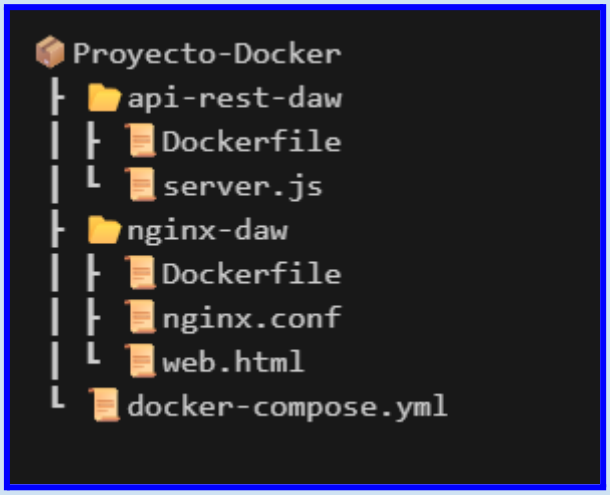
```
C:\Users\vrida\OneDrive\Escritorio\Proyecto-Docker>docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED          STATUS          PORTS
                 NAMES
674c6a718217   proyecto-docker-api-rest    "docker-entrypoint.s…"   14 minutes ago   Up 14 minutes   0.0.0.0:3000->3000/tcp, [:
:]:3000->3000/tcp   api-rest
3c08e8215a9e   proyecto-docker-web-nginx   "/docker-entrypoint.…"   14 minutes ago   Up 14 minutes   0.0.0.0:8080->80/tcp, [::]
:8080->80/tcp       web-nginx
```

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>

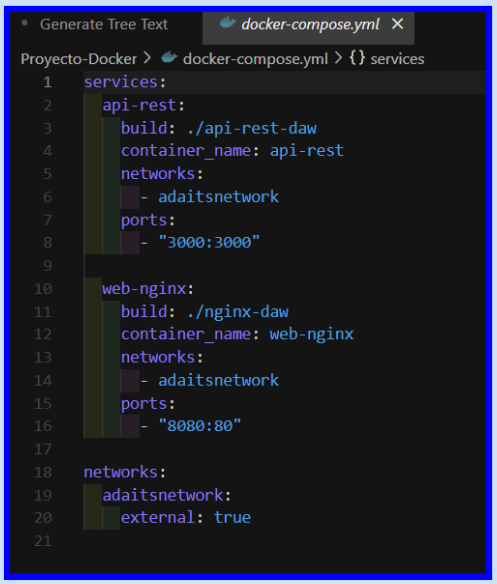
```
C:\Users\vrida\OneDrive\Escritorio\Proyecto-Docker>docker build -t api-rest-daw ./api-rest-daw
[+] Building 2.1s (10/10) FINISHED                                                                  d
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 358B
 => [internal] load metadata for docker.io/library/node:latest
 => [auth] library/node:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => 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 “adaitsnetwork”.

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

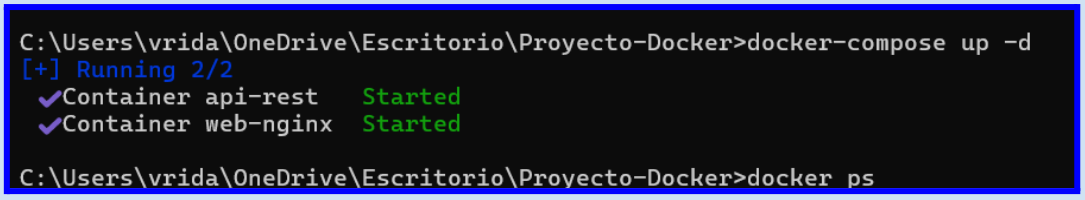
Para facilitar la creación de los contenedores, se debe realizar un docker-compose (.yml), el cual facilitará crear los servicios a la vez:

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



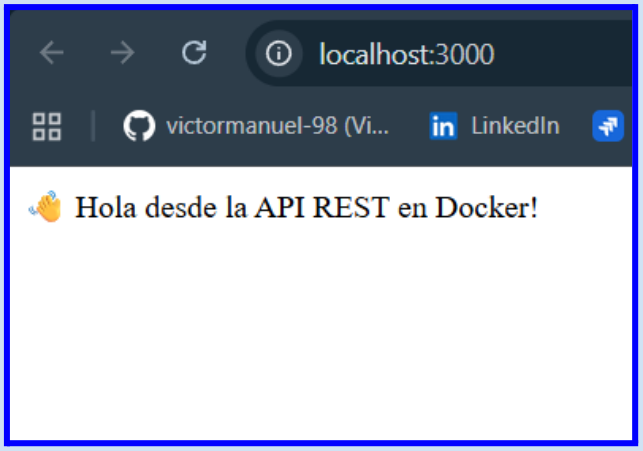
4. Despliegue de los Servicios con Docker Compose

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



- 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.