**Fundamentos y usos prácticos de Docker**

Entregable trabajo integrador: Avendaño Oscar

**Parte 1 - Conteinerizar una Aplicación**

Creando la imagen:

Ejecute el comando correspondiente para buildear la imágen. Elija un nombre de imágen y un tag acorde.

#*docker build -t mi-aplicacion:1.0 .*  
  
Muestre cuánto espacio ocupa la imaǵen una vez creada.

*# docker images*

¿Puede hacer algo para optimizar o mejorar la imágen?. Describa qué modificaciones puede hacer para optimizar la imágen.

*Usar node:alpine:  
Si el proyecto necesita solo Node.js pero no otros paquetes específicos de Alpine, se puede usar node:alpine como imagen base para simplificar instalación y reducir tamaño.*

Correr la aplicación

Una vez creada la imágen, debería ser capaz de correr la aplicación.

Ejecute un comando para poder correr la aplicación.

# *docker run -p 3000:3000 mi-aplicacion:1.0*

Explique el comando y cada parámetro enviado.

$ *docker run -p 3000:3000 primer-entregable:1.0*

docker run\*\*\*: Comando utilizado para crear y ejecutar un contenedor basado en una imagen de Docker.

-p 3000:3000\*\*\*: Esta opción asigna puertos del contenedor a puertos de la máquina host: -El primero es el puerto de la máquina host.

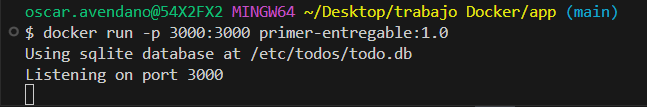
- El segundo es el puerto del contenedor.

Significa que las solicitudes enviadas al puerto `3000` de la máquina local se redirigirán al puerto `3000` dentro del contenedor.

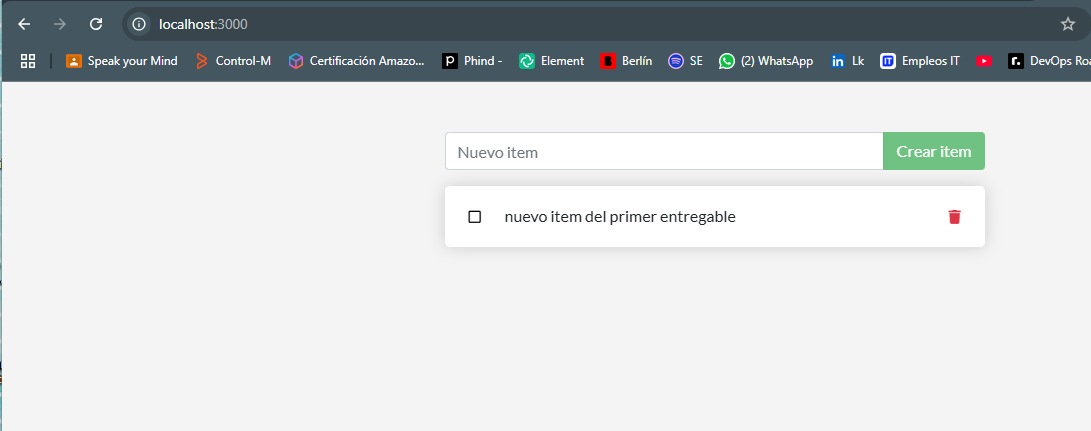
- `primer-entregable: es el nombre de la imagen.

- `1.0` es el tag de la imagen, que generalmente indica la versión de la misma.

Muestre una captura de pantalla o un copy-paste del contenedor corriendo.



Adjunte una captura de pantalla con la aplicación funcionando con la URL utilizada para acceder.



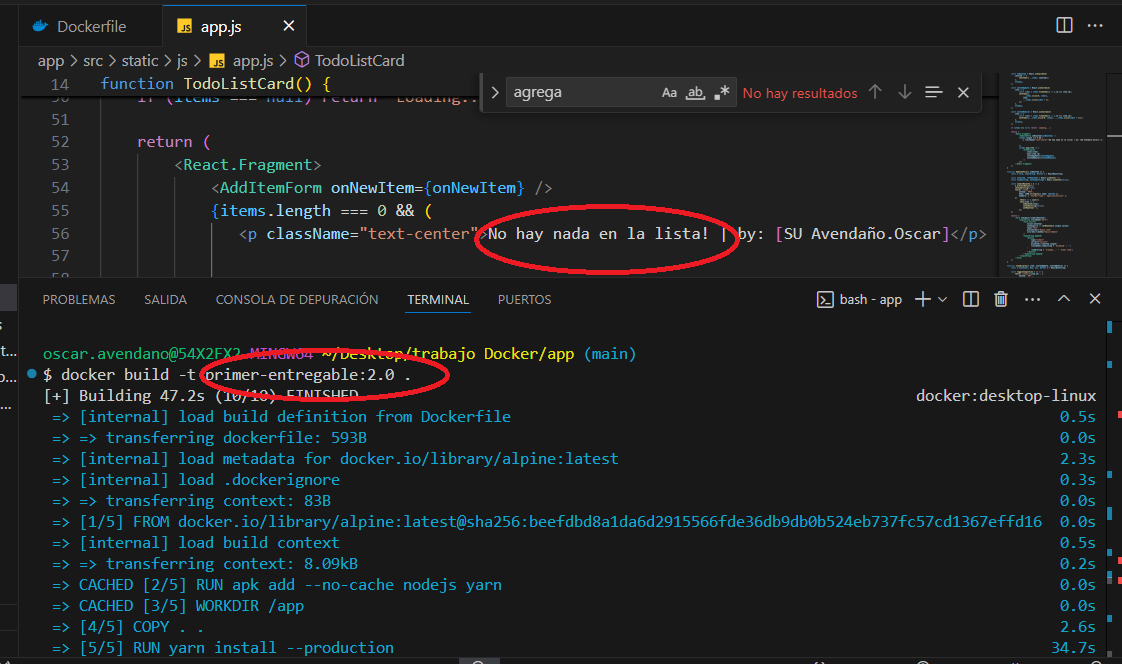
**Parte 2 - Actualizar aplicación (imágen)**

1. Actualizar el código fuente

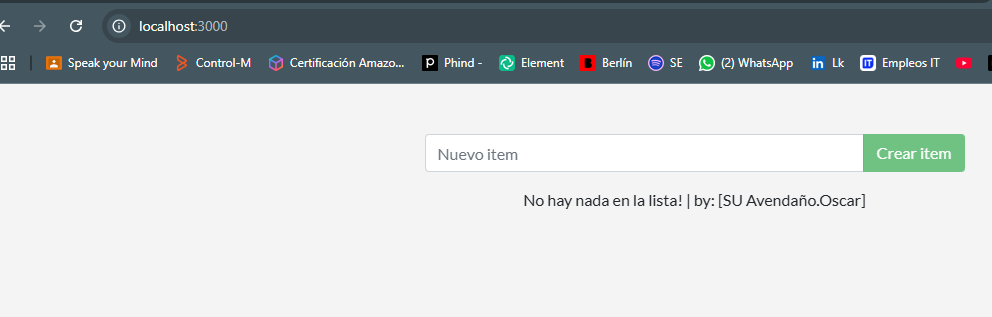
Ejecutemos los comando necesarios para que la aplicación tome los cambios. Realice un etiquetado (tag) coherente respecto a los cambios en la imágen

*# docker built -t mi-aplicacion:2.0*

Mostrar captura de pantalla con la app corriendo con las modificaciones realizadas.



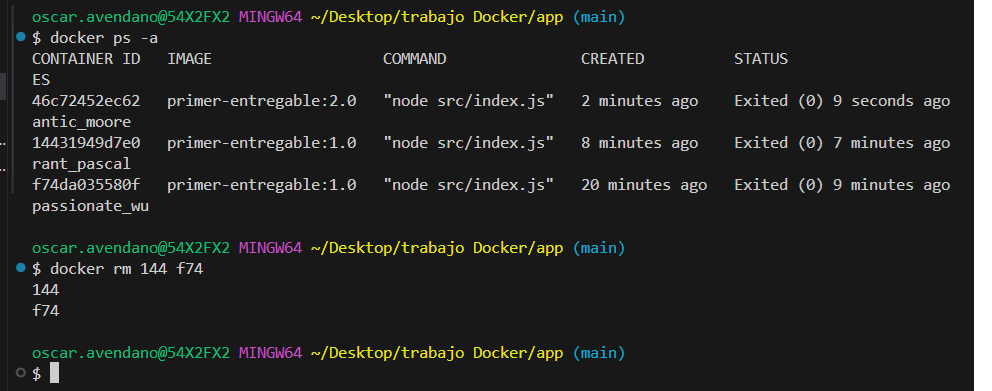
La actualizaciones realizadas, dejan a la primera versión obsoleta



2. Elimine el contenedor e imágen anterior

Elimine la imágen y el contenedor hecho en el punto anterior

*# docker rm144*



*# docker images*



**Parte 3 - Compartir app**

Para compartir la imágen de la aplicación usaremos la registry de DockerHub.

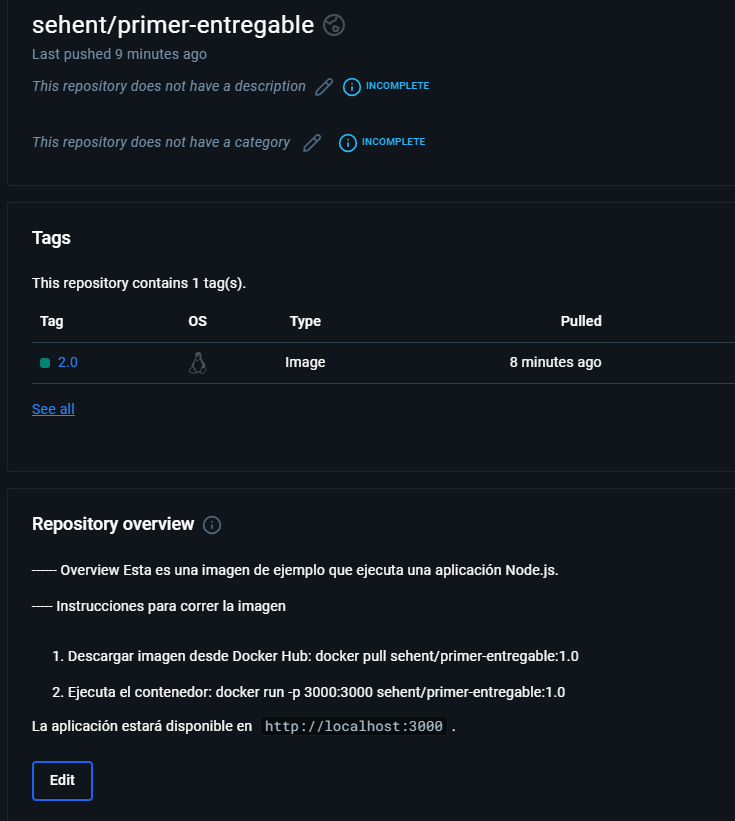
Escriba los comandos necesarios para que sea posible subir la imaǵen correctamente.

*#  docker push sehent/primer-entregable:2.0*

Comparta la URL de DockerHub para que pueda ser posible probar y descargar su imágen.

[*https://hub.docker.com/r/sehent/primer-entregable*](https://hub.docker.com/r/sehent/primer-entregable)

Agregue un overview para el repositorio de Dockerhub con instrucciones para correr la imágen y todo lo que considere necesario para que un tercero pueda ejecutar la imágen.



**Parte 4 - Persistencia de datos**

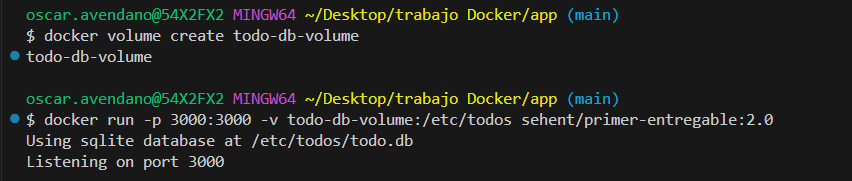
Los datos en esta APP se guardan en un archivo /etc/todos/todo.db.

Escriba los comandos utilizados para realizar lo solicitado con la explicación correspondiente.

*# $ docker volume create todo-db-volume*  
  
*# $ docker run -p 3000:3000 -v todo-db-volume:/etc/todos sehent/primer-entregable:2.0 Using sqlite database at /etc/todos/todo.db*

Decida que tipo de persistencia es la adecuada para la app.

Usé un \*\*volumen de Docker\*\* llamado `todo-db-volume` para dar persistencia de datos a la app. Monté este volumen en el directorio `/etc/todos` del contenedor, donde la aplicación guarda su archivo `todo.db`. Esto hace que los datos se mantengan incluso si el contenedor se borra. Al volver a ejecutar el contenedor con el mismo volumen, los dats previos están disponibles, lo q permite que la aplicación retome su estado anterior sin pérdida de información.de datos.



**Parte 5 - Aplicaciones multicontainer**

Crear una red para conexión entre los contenedores que servirá también para conectar a la aplicación.

*# $ docker network create app-network  
# $  docker volume create mysql-data-volume mysql-data-volume*

Crear un nuevo volumen para persistir los datos de la base MySQL. El path donde se almacenan los datos en el contenedor MySQL es /var/lib/mysql.

*# $ docker run -d \  
--name mysql-container \  
--network app-network \  
-e MYSQL\_ROOT\_PASSWORD=1234 \  
-e MYSQL\_DATABASE=todos \  
-v mysql-data-volume:/var/lib/mysql \ mysql:8.0*

Iniciar el contenedor de la aplicación utilizando el comando docker run enviando las variables de entornos necesarias para la conexión con la base de datos.

*$ docker run -d \  
--name app-container \ --network app-network \ -P 3000:3000\  
-e MYSQL\_HOST=mysql-container \  
-e MYSQL\_USER=root \  
-e MYSQL\_PASSWORD=1234 \  
-e MYSQL\_DB=todos \  
sehent/primer-entregable:2.0*

**Parte 6 - Utilizando Docker Compose**

En la carpeta raíz del proyecto, cree un archivo de docker compose compose.yml o docker-compose.yml. Adicionalmente pégue el contenido del archivo compose en este lugar:

Docker Compose es una herramienta para manejar varios contenedores a la vez con un solo archivo docker-compose.yml. En este archivo, definís servicios como la base de datos y la app, especificando cosas como las imágenes a usar, volúmenes para guardar datos y las redes que los conectan. Por ejemplo, en el docker-compose.yml la app usa una imagen de Docker Hub (sehent/mi-aplicacion:2.0), depende de la base de datos db para arrancar y se conecta a través de la red mi-red. Todo esto se levanta de una con docker compose up -d, dejando listo el entorno.

*# version: '3.8'*

*services:*

*db:*

*image: mysql:latest*

*container\_name: mi-base-datos*

*networks:*

*- mi-red*

*volumes:*

*- mysql-datos:/var/lib/mysql*

*environment:*

*MYSQL\_ROOT\_PASSWORD: mi\_contraseña\_root*

*MYSQL\_DATABASE: mi\_base\_datos*

*MYSQL\_USER: mi\_usuario*

*MYSQL\_PASSWORD: mi\_contraseña*

*app:*

*image: sehent/mi-aplicacion:2.0*

*container\_name: mi-aplicacion*

*depends\_on:*

*- db*

*networks:*

*- mi-red*

*ports:*

*- "3000:3000"*

*environment:*

*DB\_HOST: db*

*DB\_USER: mi\_usuario*

*DB\_PASSWORD: "1234"*

*DB\_NAME: mi\_base\_datos*

*networks:*

*mi-red:*

*volumes:*

*mysql-datos:*