Configuración y Creación de Contenedores

1. Configuración de una Aplicación NodeJS

- Crear el directorio del proyecto
  En tu terminal, crea un nuevo directorio para tu proyecto y navega a él:

```
mkdir secondlab
cd secondlab
```

- Inicializar nuevo proyecto Node.JS
  Inicializa el proyecto con **NPM** para crear un package.json

```
npm init -y
```

- Crea el archivo de la aplicación NodeJS
  Crea un archivo llamado `index.js` con el siguiente contenido:
  Archivo `app.js` :

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('Hola Docker!');
});

app.listen(port, () => {
  console.log(`Aplicación escuchando en http://localhost:${port}`);
});
```

- Agregar Express como dependencia
  Añade Express, un framework de servidor web para NodeJS:

```
npm install express --save
```

2. Dockerfile para la Aplicación NodeJS

Utiliza el siguiente `Dockerfile` para construir una imagen de esta aplicación.

Archivo `Dockerfile` :

```
FROM node:latest

WORKDIR /usr/src/app

COPY package*.json ./
RUN npm install

COPY . .

EXPOSE 3000

CMD [ "node", "index.js" ]
```

3. Creación de Contenedores

Ahora, construye la imagen y crea un contenedor con ella.

Construir la Imagen:

```
docker build -t lab2 .
```

Crear y Ejecutar el Contenedor:

```
docker run -d -p 3000:3000 lab2
```

Manejo de Logs de Contenedores

1. **Visualizar Logs**:

   ```
   docker logs [CONTAINER_ID]
   ```

2. **Seguimiento de Logs en Tiempo Real** (`--follow`):

   ```
   docker logs -f [CONTAINER_ID]
   ```

3. **Limitar la Cantidad de Líneas de Logs** (`--tail`):

   ```
   docker logs --tail 10 [CONTAINER_ID]
   ```

4. **Mostrar Logs Desde un Tiempo Específico** (`--since`):

```
docker logs --since 2023-01-01T10:00:00 [CONTAINER_ID]
```

> **Nota:** puedes reemplazar la fecha y hora a la deseada

5. **Mostrar Logs Hasta un Tiempo Específico** (`--until`):

```
docker logs --until 2023-01-01T11:00:00 [CONTAINER_ID]
```

> **Nota:** puedes reemplazar la fecha y hora a la deseada

6. **Mostrar Timestamps en los Logs** (`--timestamps`):

```
docker logs --timestamps [CONTAINER_ID]
```

Interacción Avanzada con Contenedores

1. Uso de `docker exec`

El comando `docker exec` se utiliza para ejecutar comandos dentro de un contenedor que ya está en ejecución.

- **Ejecutar un comando interactivo** (por ejemplo, bash) en un contenedor:

```
docker exec -it [CONTAINER_ID] /bin/bash
```

2. Uso de `docker cp`

El comando `docker cp` permite copiar archivos entre un contenedor y el host local.

- **Copiar un archivo del host al contenedor**:

```
docker cp [FILE_PATH] [CONTAINER_ID]:[CONTAINER_PATH]
```

> **Ejemplo** de un archivo llamado holamundo.txt en el escritorio y copiarlo a raiz del contenedor

```
docker cp C:\Users\tu_usuario\Desktop\holamundo.txt <nombre-o-id-del-contenedor>:/
```

- **Copiar un archivo del contenedor al host**:

```
docker cp [CONTAINER_ID]:[CONTAINER_FILE_PATH] [HOST_PATH]
```

3. Limitación de Recursos

Docker permite limitar los recursos de CPU y memoria de un contenedor.

- **Limitar la memoria**:

```
docker run -d -p 3000:3000 --memory="1g" lab2
```

- **Limitar el uso de CPU** (por ejemplo, limitar a 0.5 CPUs):

```
docker run -d -p 3000:3000 --cpus="0.5" lab2
```

Monitoreo y Manipulación de Imágenes

1. Monitoreo de Contenedores con `docker stats`

Este comando proporciona una transmisión en vivo de los recursos utilizados por los contenedores.

- **Monitorear todos los contenedores**:

```
docker stats
```

- **Monitorear un contenedor específico**:

```
docker stats [CONTAINER_ID]
```

2. Creación de Imágenes mediante Commit

Puedes crear una nueva imagen a partir de un contenedor modificado.

- **Modificar un contenedor y luego crear una imagen**:

```
docker commit [CONTAINER_ID] lab2-modificada
```

3. Exportación de Contenedores

Exporta un contenedor a un archivo tar.

- **crear contenedor httpd**

```
docker run -d -p 8080:80 --name httpdexport httpd
```

- **Exportar un contenedor**:

```
docker export httpdexport > container.tar
```

Importar y Ejecutar un Contenedor Exportado

1. Importar un Contenedor desde un Archivo Tar

Una vez que hayas exportado un contenedor a un archivo tar, puedes importarlo de nuevo para crear una imagen.

- **Importar un archivo tar como imagen**:

```
docker import container.tar my-imported-container
```

2. Ejecutar un Contenedor desde la Imagen Importada

Ahora, puedes crear y ejecutar un contenedor a partir de la imagen importada.

- **Ejecutar un contenedor a partir de la imagen importada**:

```
docker run -d -p 3000:80 my-imported-container
```

> **Nota** este comando nos dará un error intente descubrir el error