Actividad Linea III

1. ¿Qué es un archivo YAML y para que se utiliza en desarrollo de software?

   YAML (YAML Ain't Markup Language) es un formato de serialización de datos que se utiliza para representar información estructurada de una manera legible para los humanos. Es muy popular en el desarrollo de software debido a su simplicidad y claridad.

   Uso en desarrollo de software:

   Configuración: Es utilizado para almacenar configuraciones de aplicaciones y servicios (por ejemplo, archivos docker-compose.yml para definir servicios en Docker).

   Intercambio de datos: Para transmitir datos entre diferentes sistemas de forma legible y estructurada.

   Definición de Infraestructura como Código: En herramientas de infraestructura como Terraform, Ansible y Kubernetes, se utilizan archivos YAML para definir recursos y configuraciones.

2. ¿Diferencia entre YAML y JSON?

Característica	**YAML**	**JSON**
Legibilidad	Más legible para los humanos debido a su sintaxis simple (sin comillas ni llaves).	Menos legible para los humanos debido a la sintaxis más estricta (uso de comillas, llaves y comas).
Tipos de Datos	Soporta fechas, referencias, y datos complejos de manera más natural.	Solo admite tipos básicos (números, cadenas, objetos, arrays).
Sintaxis	Usa indentación para definir estructuras.	Usa llaves {} y corchetes [] para definir estructuras.
Comentarios	Permite comentarios usando #.	No permite comentarios.
Uso Común	Configuración de aplicaciones, archivos de despliegue, herramientas de infraestructura.	APIs, configuraciones ligeras, intercambio de datos

3. ¿Cuándo se recomienda usar uno sobre el otros YAML y JSON ?

   Cuando usar YAML:

   - ✓ Se necesita una representación de datos más legible para los humanos.
   - ✓ Se están gestionando configuraciones complejas o archivos de infraestructura.
   - ✓ Se quiere hacer uso de comentarios en la configuración.

   Cuando usar JSON:

   - ✓ El intercambio de datos es con aplicaciones o servicios que requieren un formato ligero y fácil de parsear.
   - ✓ Se está trabajando con APIs o bases de datos que utilizan JSON.
   - ✓ Se necesita un formato estandarizado y más eficiente en términos de procesamiento (JSON es más fácil de manejar en muchos lenguajes de programación).

4. ¿Cómo se crean los archivos yml ?

   Para crear un archivo .yml simplemente se necesita un editor de texto y guardarlo con la extensión .yml o .yaml. Algunos ejemplos de editores de texto incluyen:

   - ✓ Visual Studio Code (con soporte para sintaxis YAML).
   - ✓ Sublime Text.
   - ✓ Atom.
   - ✓ Vim o Emacs.

   Este archivo YAML define un servicio llamado web que utiliza la imagen de Docker nginx y mapea el puerto 80 del contenedor al puerto 8080 en la máquina host.

   Ejemplo de un archivo .yml:

```
version: '3'

services:

    web:

        image: nginx

        ports:

            - "8080:80"
```

5. Crear una BD que tenga 5 tablas y desplegar el servicio usando archivos yml
   Crear un contenedor
   Crear el volumen
   crear la base de datos Supermercado
   con los siguientes tablas

   Usuarios
   Id_usuario,
   nombre VARCHAR(100),
   email VARCHAR(100)

   Productos
   Id_producto,
   nombre VARCHAR(100),
   precio DECIMAL

   Pedidos
   Id_pedido,
   id_usuario
   fecha TIMESTAMP

   DetallesPedido
   Id_detallepedido,
   id_pedido INT
   id_producto INT
   cantidad INT

   Categorías
   Id_categoria,
   nombre VARCHAR(100)

Comandos Comunes de Docker Compose

1. **Crear y levantar el contenedor → docker-compose up -d**
   - **Descripción**: Inicia los servicios definidos en el archivo **docker-compose.yml**. Si los contenedores no existen, los crea. También puede construir imágenes si es necesario.
   - **Contras**:
     - Puede ser lento si hay que construir imágenes grandes.
     - Si no se usa el flag **-d**, la salida del contenedor se mostrará en la terminal, lo que puede ser difícil de manejar si hay muchos contenedores.

- Si hay un error en la configuración, puede ser difícil de depurar.

2. **Parar el contenedor → docker-compose down**
    - **Descripción**: Detiene y elimina los contenedores, redes y volúmenes definidos en el archivo **docker-compose.yml**.
    - **Contras**:
        - Elimina todos los datos en los volúmenes a menos que se use el flag **-v**, lo que puede llevar a la pérdida de datos si no se tiene cuidado.
        - Puede ser confuso si se tienen múltiples entornos de desarrollo y producción.
3. **docker-compose build**
    - **Descripción**: Construye las imágenes de los servicios definidos en el archivo **docker-compose.yml**.
    - **Contras**:
        - Puede ser lento si hay muchas dependencias o si las imágenes son grandes.
        - No se ejecuta automáticamente al usar **docker-compose up** si no se especifica el flag **--build**.
4. **docker-compose logs**
    - **Descripción**: Muestra los logs de los contenedores en ejecución.
    - **Contras**:
        - Puede ser difícil de leer si hay muchos contenedores generando logs.
        - No permite filtrar logs de manera efectiva sin herramientas adicionales.
5. **docker-compose exec**
    - **Descripción**: Ejecuta un comando en un contenedor en ejecución.
    - **Contras**:
        - Si el contenedor no está en ejecución, el comando fallará.
        - Puede ser confuso si se ejecutan múltiples instancias del mismo servicio.
6. **docker-compose ps**
    - **Descripción**: Muestra el estado de los contenedores en ejecución.
    - **Contras**:
        - No proporciona información detallada sobre los logs o el estado interno de los contenedores.

- Puede ser confuso si hay muchos contenedores en ejecución.

7. **docker-compose pull**
    - **Descripción**: Descarga las imágenes de los servicios definidos en el archivo **docker-compose.yml**.
    - **Contras**:
        - Puede ser lento si hay muchas imágenes o si la conexión a Internet es lenta.
        - No actualiza automáticamente las imágenes si ya están descargadas.
8. **docker-compose rm**
    - **Descripción**: Elimina contenedores detenidos.
    - **Contras**:
        - Puede llevar a la pérdida de datos si no se han guardado correctamente.
        - No elimina volúmenes a menos que se use el flag **-v**.

Consultas:

```
UPDATE Productos SET precio = 1.10 WHERE nombre = 'Pan';

-- Borrar (DELETE)

DELETE FROM Productos WHERE nombre = 'Leche';


-- INNER JOIN (unir tablas con registros que tienen coincidencias)

SELECT p.nombre, u.nombre, d.cantidad

FROM Pedidos p

JOIN DetallesPedido d ON p.Id_pedido = d.Id_pedido

JOIN Usuarios u ON p.Id_usuario = u.Id_usuario;


-- LEFT JOIN (unir tablas mostrando todos los registros de la tabla izquierda)

SELECT p.nombre, u.nombre, d.cantidad

FROM Pedidos p

LEFT JOIN DetallesPedido d ON p.Id_pedido = d.Id_pedido

LEFT JOIN Usuarios u ON p.Id_usuario = u.Id_usuario;
```

```
-- RIGHT JOIN (unir tablas mostrando todos los registros de la tabla derecha)
SELECT p.nombre, u.nombre, d.cantidad
FROM Pedidos p
RIGHT JOIN DetallesPedido d ON p.Id_pedido = d.Id_pedido
RIGHT JOIN Usuarios u ON p.Id_usuario = u.Id_usuario;
```