

Taller Clase

Juan Felipe Tarazona Pita ID:832837

Corporación Universitaria Minuto de Dios

Ingeniería de Sistemas, 7° Semestre

Desarrolla Basado en Plataformas NRC:66168

Pr. Alexander Matallana Porras

20 de marzo 2025

Taller Clase

¿Qué es un archivo yaml?

Un YAML es un formato de serialización de datos diseñado para ser fácil de leer y escribir por humanos. Se usa comúnmente en archivos de configuración, intercambio de datos y almacenamiento estructurado. Aunque su nombre originalmente significaba "Yet Another Markup Language" (Otro lenguaje de marcado más), actualmente se define como "YAML Ain't Markup Language" (YAML no es un lenguaje de marcado), destacando que su propósito es representar datos de manera clara y simple, en lugar de documentos con formato.

De manera mas simple el YAML (o YML) es un formato de texto que se usa para organizar información de manera sencilla y clara. Sirve principalmente para archivos de configuración en aplicaciones y servidores, intercambio de datos entre sistemas y automatización en herramientas como Docker y GitHub Actions. Es fácil de leer porque usa sangrías en lugar de llaves o etiquetas, como JSON o XML.

¿Qué es un archivo json?

JSON (JavaScript Object Notation) es un formato de texto utilizado para estructurar y transmitir datos entre sistemas de manera sencilla y ligera. Es muy común en aplicaciones web y móviles, ya que permite que un servidor y un cliente intercambien información sin necesidad de recargar la página, como en solicitudes AJAX.

Aunque su origen se remonta al año 2000, JSON se incorporó oficialmente a JavaScript con ECMAScript 5 y hoy es compatible con casi todos los lenguajes de programación. Su popularidad se debe a que es más fácil de leer y manejar que XML, lo que lo convierte en una opción eficiente para almacenar y compartir datos.

Diferencia entre un yml y un json

JSON y YAML son dos formatos utilizados para almacenar y transmitir datos, pero tienen diferencias importantes.

YAML es más fácil de leer porque usa indentación en lugar de símbolos como llaves {} y comas,. Además, permite agregar comentarios con #, lo que facilita la documentación dentro del archivo. Por esta razón, se usa mucho en archivos de configuración, como en Docker, Kubernetes y GitHub Actions.

JSON, en cambio, es más estructurado y estricto, ya que utiliza llaves {} para organizar los datos y no permite comentarios nativos. Es muy popular en la transmisión de datos entre sistemas, especialmente en APIs y bases de datos.

En cuanto al tamaño, YAML suele ser más compacto y limpio, mientras que JSON puede ocupar más espacio debido a los caracteres adicionales.

Si necesitas un archivo de configuración fácil de leer, YAML es una mejor opción. Si trabajas con APIs o necesitas intercambiar datos entre sistemas, JSON es más adecuado.

¿para que sirve el Docker-compose.yml?

Docker Compose es una herramienta que permite definir y administrar aplicaciones con múltiples contenedores en Docker mediante un solo archivo de configuración llamado `docker-compose.yml`. Este archivo se utiliza para describir los servicios, redes y volúmenes que componen una aplicación, facilitando su despliegue y administración.




Por ejemplo, un `docker-compose.yml` puede definir un entorno con una base de datos PostgreSQL y una aplicación en Node.js, especificando cómo deben interactuar entre sí. En el archivo, se pueden definir aspectos como el nombre del servicio, la imagen de Docker que se usará, los volúmenes para persistencia de datos, variables de entorno, y la configuración de red.

Una de sus principales ventajas es la capacidad de levantar toda la infraestructura con un solo comando (`docker-compose up`), lo que simplifica el proceso de desarrollo y despliegue. Además, permite escalabilidad definiendo múltiples réplicas de un servicio y facilita la integración con entornos de producción, como Kubernetes.

En resumen, el docker-compose.yml es una pieza clave para la automatización y gestión eficiente de aplicaciones basadas en contenedores, permitiendo configurar entornos complejos de manera declarativa y reproducible.

¿Como se crea un contenedor usando yml?

En una carpeta, creamos un archivo de texto (.txt), luego cambiamos su extensión a .yml para convertirlo en un archivo de configuración de Docker Compose. Posterior a eso en un editor de código procedemos a crear nuestro contenedor , donde especificamos todos los atributos.

	Base_Pokemon	20/03/2025 2:48 p. m.	Carpeta de archivos	
	docker-compose.yml	20/03/2025 2:28 p. m.	Archivo de origen ...	1 KB
	init.sql	20/03/2025 12:41 p. m.	SQL Text File	2 KB

```

> Run All Services
services:
  > Run Service
  mysql:
    image: mysql:latest
    container_name: "mi_mysql"
    restart: "always"
    environment:
      MYSQL_ROOT_PASSWORD: "FeNrIr2233@"
      MYSQL_DATABASE: "Base_Pokemon"
      MYSQL_USER: "usuario"
      MYSQL_PASSWORD: "FeNrIr2233@"
    ports:
      - "3307:3306"
    volumes:
      - "C:/Users/USUARIO/Desktop/ejercicioBD/compose:/var/lib/mysql"
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
volumes:
  mysql_data:

```



¿Cómo se levanta el Docker -compose .yaml?

Para levantar un archivo docker-compose.yml, primero debemos abrir una terminal en la carpeta donde se encuentra el archivo de configuración del proyecto. Una vez ubicados en el directorio correcto, utilizamos el comando `docker-compose up -d`. Este comando inicia los contenedores definidos en el archivo y los ejecuta en segundo plano, permitiendo que la aplicación funcione sin bloquear la terminal. Esto facilita el despliegue y la gestión de los servicios, asegurando que el entorno configurado en Docker Compose se inicie correctamente.

¿Crea una base de datos se crea un contenedor con 4 tablas e insertar registros?

Primero, creamos una carpeta llamada EjercicioBD. Dentro de esta, generamos dos archivos: `docker-compose.yml` e `init.sql`. Para hacerlo de forma rápida, podemos crear un archivo temporal con la extensión `.txt` y luego cambiarle el nombre y la extensión a los que necesitamos.



 docker-compose.yml	21/03/2025 2:14 p. m.	Archivo de origen ...	1 KB
 init.sql	20/03/2025 12:41 p. m.	SQL Text File	2 KB

Una vez creados los archivos, abrimos docker-compose.yml en Visual Studio Code. Luego, definimos la estructura YAML para configurar un contenedor con volumen y la base de datos.

```

  Run All Services
services:
  Run Service
  mysql:
    image: mysql:latest
    container_name: "mi_mysql"
    restart: "always"
    environment:
      MYSQL_ROOT_PASSWORD: "FeNrIr2233@"
      MYSQL_DATABASE: "Base_Pokemon"
      MYSQL_USER: "usuario"
      MYSQL_PASSWORD: "FeNrIr2233@"
    ports:
      - "3307:3306"
    volumes:
      - "C:/Users/USUARIO/Desktop/ejercicioBD/compose:/var/lib/mysql"
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
volumes:
  mysql_data:
```

Luego, abrimos init.sql para definir las tablas de la base de datos e insertar tres registros en cada una. Primero, escribimos los datos en un archivo .txt y luego cambiamos la extensión a .sql. Finalmente, verificamos en MySQL que las tablas y los registros se hayan creado correctamente.

```

Juan Felipe Tarazona Pita  desktop.ini  docker-compose.yml.txt  init.sql
Archivo  Editar  Ver

CREATE DATABASE IF NOT EXISTS Base_Pokemon;
USE Base_Pokemon;

CREATE TABLE pokemon (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  tipo VARCHAR(50) NOT NULL,
  nivel INT NOT NULL
);

CREATE TABLE entrenadores (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  ciudad VARCHAR(100) NOT NULL
);

CREATE TABLE batallas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  entrenador1_id INT,
  entrenador2_id INT,
  ganador_id INT,
  fecha DATE,
  FOREIGN KEY (entrenador1_id) REFERENCES entrenadores(id),
  FOREIGN KEY (entrenador2_id) REFERENCES entrenadores(id),
  FOREIGN KEY (ganador_id) REFERENCES entrenadores(id)
);

CREATE TABLE equipo_pokemon (
  id INT AUTO_INCREMENT PRIMARY KEY,
  entrenador_id INT,
  pokemon_id INT,
  FOREIGN KEY (entrenador_id) REFERENCES entrenadores(id),
  FOREIGN KEY (pokemon_id) REFERENCES pokemon(id)
);

-- Insertar Pokémons
INSERT INTO pokemon (nombre, tipo, nivel) VALUES ('Pikachu', 'Eléctrico', 25);
INSERT INTO pokemon (nombre, tipo, nivel) VALUES ('Charmander', 'Fuego', 20);
INSERT INTO pokemon (nombre, tipo, nivel) VALUES ('Squirtle', 'Agua', 22);
INSERT INTO pokemon (nombre, tipo, nivel) VALUES ('Bulbasaur', 'Planta', 23);

-- Insertar Entrenadores
INSERT INTO entrenadores (nombre, ciudad) VALUES ('Ash Ketchum', 'Pueblo Paleta');
INSERT INTO entrenadores (nombre, ciudad) VALUES ('Misty', 'Ciudad Celeste');
INSERT INTO entrenadores (nombre, ciudad) VALUES ('Brock', 'Ciudad Plateada');

Ln 54, Col 1  1.867 caracteres.  100%  Windows (CRLF)  UTF-8

```

Limit to 1000 rows

```

1 • CREATE DATABASE IF NOT EXISTS Base_Pokemon;
2 • USE Base_Pokemon;
3
4 • CREATE TABLE pokemon (
5   id INT AUTO_INCREMENT PRIMARY KEY,
6   nombre VARCHAR(100) NOT NULL,
7   tipo VARCHAR(50) NOT NULL,
8   nivel INT NOT NULL
9 );
10
11 • CREATE TABLE entrenadores (
12   id INT AUTO_INCREMENT PRIMARY KEY,
13   nombre VARCHAR(100) NOT NULL,
14   ciudad VARCHAR(100) NOT NULL
15 );
16
17 • CREATE TABLE batallas (
18   id INT AUTO_INCREMENT PRIMARY KEY,
19   entrenador1_id INT,
20   entrenador2_id INT,
21   ganador_id INT,
22   fecha DATE,
23   FOREIGN KEY (entrenador1_id) REFERENCES entrenadores(id),
24   FOREIGN KEY (entrenador2_id) REFERENCES entrenadores(id),

```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Context Help Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
---	------	--------	---------	------------------

Nos ubicamos en la carpeta donde están docker-compose.yml e init.sql, abrimos la terminal y ejecutamos el comando docker-compose up -d para iniciar el contenedor y levantar la base de datos.

```

C:\Windows\System32\cmd.e
Microsoft Windows [Versión 10.0.26100.3476]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\USUARIO\Desktop\ejercicioBD>cd compose

C:\Users\USUARIO\Desktop\ejercicioBD\compose>docker-compose up -d
unable to get image 'mysql:latest': error during connect: Get "http://%2F%2F.%2Fpipe%2FdockerDesktopLinuxEngine/v1.48/images/mysql:latest/json": open //.pipe/dockerDesktopLinuxEngine: The system cannot find the file specified.

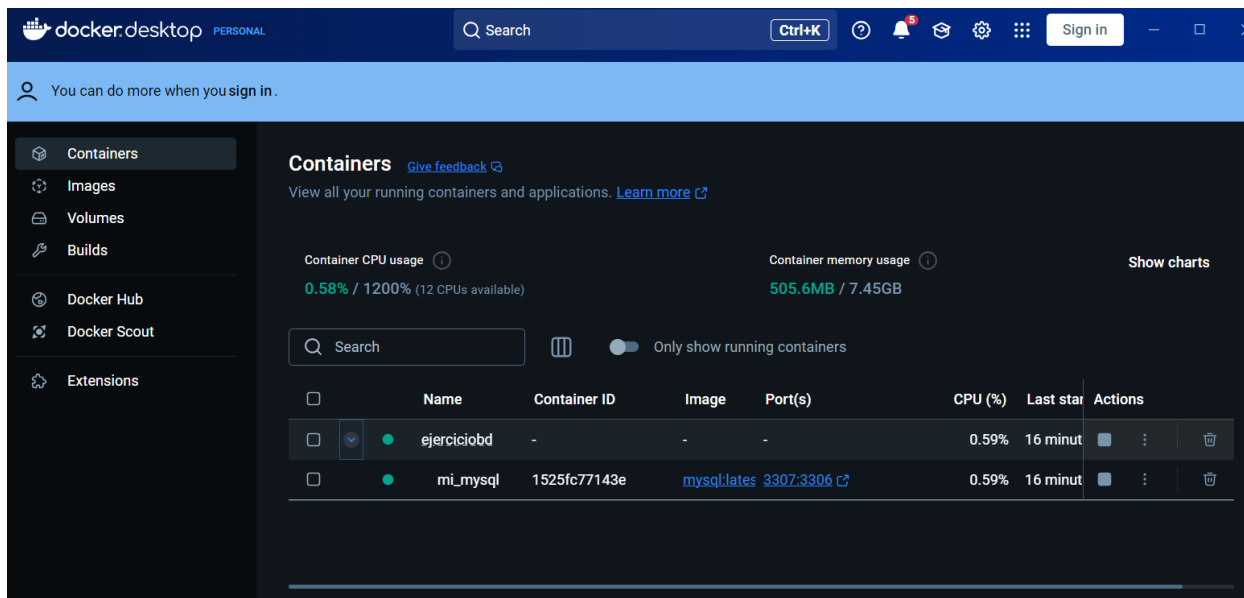
C:\Users\USUARIO\Desktop\ejercicioBD\compose>docker-compose up -d
[+] Running 0/1
  - Container mi_mysql Creating 0.1s
Error response from daemon: Conflict. The container name "/mi_mysql" is already in use by container "13843b93fc20aa49a9ff930a8102af99564833b9047515a99642325c8bb54a4a". You have to remove (or rename) that container to be able to reuse that name.

C:\Users\USUARIO\Desktop\ejercicioBD\compose>docker-compose up -d
[+] Running 1/1
  ✓ Container mi_mysql Started 0.6s

C:\Users\USUARIO\Desktop\ejercicioBD\compose>

```

Verificamos en Docker Desktop que el contenedor esté en ejecución.



Nos dirigimos a la carpeta del proyecto y revisamos si, dentro de la carpeta con el nombre de la base de datos, se guardaron las tablas creadas en init.sql, también apreciamos que se creo el volumen y demás.

compose	20/03/2025 2:30 p. m.	Carpeta de archivos	
docker-compose.yml	20/03/2025 2:28 p. m.	Archivo de origen ...	1 KB
init.sql	20/03/2025 12:41 p. m.	SQL Text File	2 KB

#innodb_redo	20/03/2025 2:30 p. m.	Carpeta de archivos	
#innodb_temp	20/03/2025 2:30 p. m.	Carpeta de archivos	
Base_Pokemon	20/03/2025 2:30 p. m.	Carpeta de archivos	
mysql	20/03/2025 2:30 p. m.	Carpeta de archivos	
performance	20/03/2025 2:30 p. m.	Carpeta de archivos	
sys	20/03/2025 2:30 p. m.	Carpeta de archivos	
#ib_16384_0.dblwr	20/03/2025 2:30 p. m.	Archivo DBLWR	6.144 KB
#ib_16384_1.dblwr	20/03/2025 2:29 p. m.	Archivo DBLWR	14.336 KB
auto.cnf	20/03/2025 2:29 p. m.	Archivo CNF	1 KB

ejercicioBD > compose > Base_Pokemon				
Ordenar Ver ...				
Nombre	Fecha de modificación	Tipo	Tamaño	
batallas.ibd	20/03/2025 2:30 p. m.	Archivo IBD	160 KB	
entrenadores.ibd	20/03/2025 2:30 p. m.	Archivo IBD	112 KB	
equipo_pokemon.ibd	20/03/2025 2:30 p. m.	Archivo IBD	144 KB	
pokemon.ibd	20/03/2025 2:30 p. m.	Archivo IBD	112 KB	

Finalmente, abrimos MySQL Workbench, realizamos la conexión con la base de datos y verificamos que los registros se hayan guardado correctamente. Para ello, utilizamos ingeniería inversa para inspeccionar la estructura y el contenido de la base de datos.

Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.


[Browse Documentation >](#)[Read the Blog >](#)[Discuss on the Forums >](#)

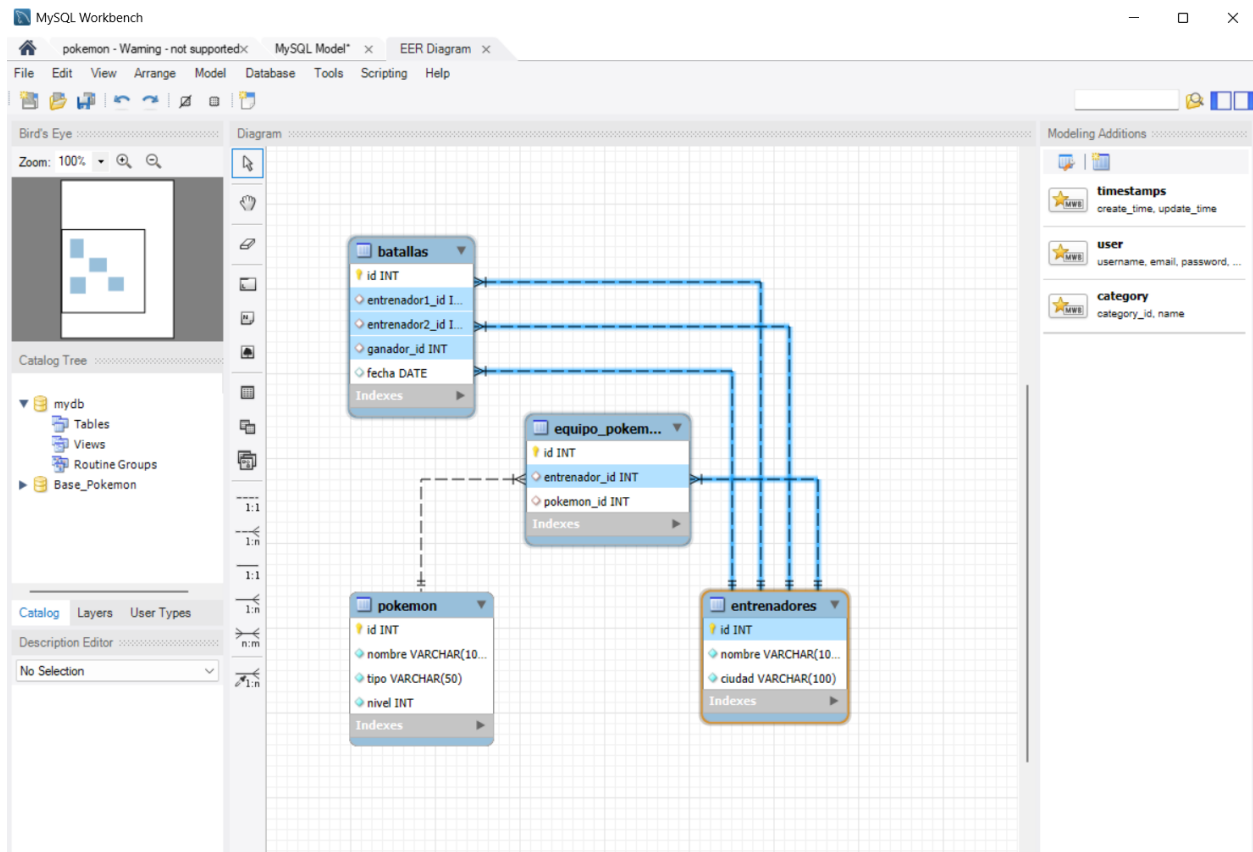
MySQL Connections

 Filter connections

Local instance MySQL80

 root

 localhost:3306



Referencias

de Imagina, E. (2025, marzo 22). *Qué es Docker Compose y Cómo Usarlo*.

Imaginaformacion.com; Imagina Formación. <https://imaginaformacion.com/tutoriales/que-es-docker-compose>

de Souza, I. (2021, agosto 19). *Archivo JSON: ¿qué es y para qué sirve en las páginas web?* Rock Content - ES; Rock Content. <https://rockcontent.com/es/blog/archivo-json/>

IBM Product Master 12.0.0. (2025, enero 7). Ibm.com. <https://www.ibm.com/docs/es/product-master/12.0.0?topic=images-starting-accessing-closing-docker-containers>

YAML: qué es, usos, sintaxis y ejemplos. (s/f). Redhat.com. Recuperado el 22 de marzo de 2025, de <https://www.redhat.com/es/topics/automation/what-is-yaml>