

Taller

Jefferson Duban Parra ID:827888

Corporación Universitaria Minuto de Dios

Ingeniería de Sistemas, 7° Semestre

Desarrolla Basado en Plataformas NRC:66168

Pr. Alexander Matallana Porras

20 de marzo 2025

1. Archivos Yml

Es un formato de serialización de datos diseñado para ser fácil de leer y escribir para humanos. Es ampliamente usado en la configuración de aplicaciones, infraestructura como código (IaC), y herramientas como Docker, Kubernetes, Ansible y CI/CD.

Características

- Usa indentación en lugar de llaves {} o corchetes [].
- Permite comentarios con #.
- Admite listas (- elemento) y diccionarios (clave: valor).
- Soporta tipos de datos como strings, números, booleanos y nulos.

1.1 Archivos Json

Es un formato ligero de intercambio de datos basado en texto. Fue diseñado para ser fácilmente leído por humanos y procesado por máquinas. Se deriva de la sintaxis de los objetos en JavaScript, aunque es independiente del lenguaje.

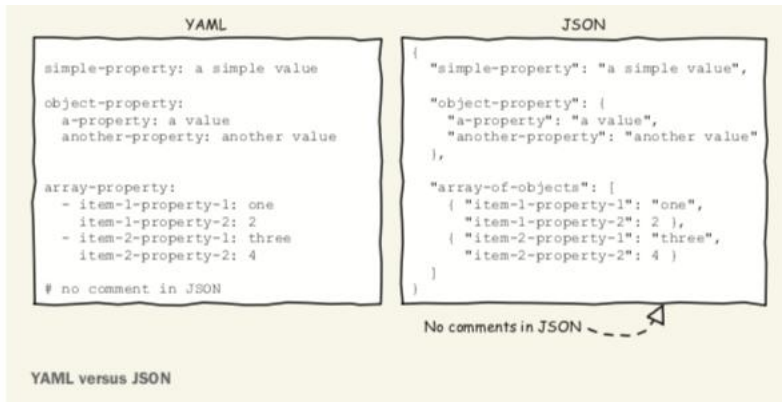
Características

- Utiliza una estructura basada en pares clave-valor { "clave": "valor" }.
- Puede representar objetos ({}), listas ([]), cadenas de texto, números, booleanos y valores null.
- Es ampliamente utilizado en APIs web, bases de datos NoSQL (como MongoDB), configuración de aplicaciones y almacenamiento de datos.
- No permite comentarios.

1.2 Json Vs Archivos Yml

Ambos son formatos de serialización de datos utilizados para intercambiar información estructurada entre aplicaciones. JSON está más enfocado en la compatibilidad con sistemas, mientras que YAML prioriza la legibilidad humana.

Característica	Json	Yml
Tipos de datos	Números, booleanos, nulos, cadenas, listas, objetos	Admite más tipos de datos anidados
Uso	APIs, bases de datos, aplicaciones web	Configuración en DevOps, Docker, Kubernetes, CI/CD
Control de versiones	Sí, pero no es tan sencillo analizar y comprender de un solo vistazo las diferencias entre las versiones.	Sí, y es fácil analizar y comprender de un solo vistazo las diferencias entre las versiones.



2. Docker -compose.yml uso

docker-compose.yml es un archivo de configuración en formato YAML que se usa con Docker Compose para definir y administrar múltiples contenedores en una aplicación.

- Permite orquestar múltiples contenedores con una sola configuración.
- Facilita la automatización del despliegue de servicios interconectados.
- Mejora la reproducibilidad y el mantenimiento de entornos de desarrollo.
- Define volúmenes, redes y variables de entorno de manera sencilla.

3. Como se crea un contenedor usando yaml

Para crear un contenedor usamos la siguiente estructura, donde podemos encontrar el servicio de MSQL, también el nombre de la base de datos, el password, declaramos el usuario, y los puertos.

```
docker-compose.yml X
C: > Users > Lenovo > Desktop > Supermecado > docker-compose.yml
1
2  version: '3.8'
3
4  services:
5    mysql:
6      image: mysql:latest
7      container_name: Supermecado
8      restart: always
9      environment:
10       MYSQL_ROOT_PASSWORD: "1234"
11       MYSQL_DATABASE: "Super"
12       MYSQL_USER: "user"
13       MYSQL_PASSWORD: "password"
14
15     ports:
16       - "3307:3306"
17
18     volumes:
19       - "/c/Users/Lenovo/Desktop/Supermecado/Supermecado_yml/Taller:/var/lib/mysql"
20       - "/init.sql:/docker-entrypoint-initdb.d/init.sql"
21
22
23
24
25
```

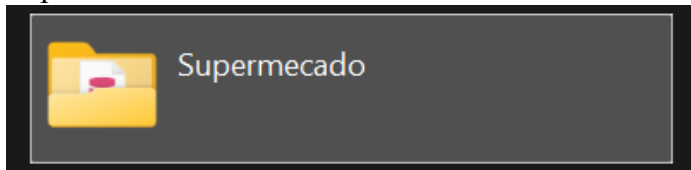
4. Como se hace para levantar el Docker -compose .yaml

para levantar usamos el comando Docker-compose up -d en la terminal donde esté enlazada la carpeta del proyecto que se está realizando.



5. Crea una base de datos se crea un contenedor con 4 tablas e insertar registros.

Paso a paso.

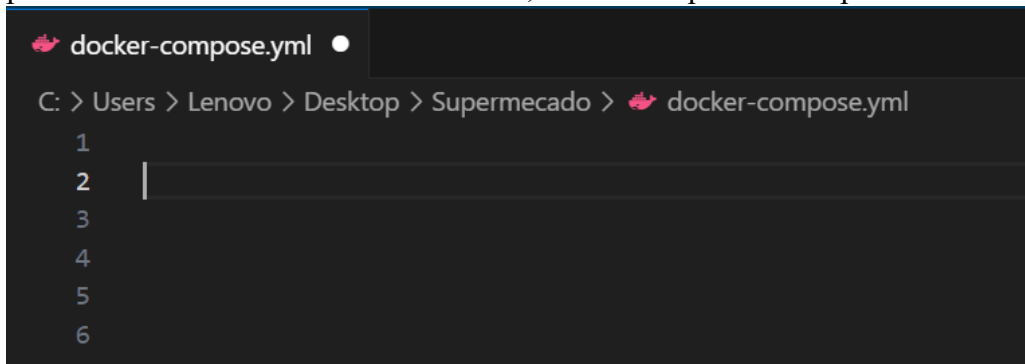
1. Creamos una carpeta con el nombre del proyecto en este caso se llama Supermercado.



2. Dentro de la carpeta creamos dos archivos nuevos.
El primero es Docker-compose.yaml y el segundo init.sql

 docker-compose.yaml	20/03/2025 1:08 p.m.	Archivo de origen Ya...	1 KB
 init.sql	20/03/2025 2:17 p.m.	Archivo de origen SQL	2 KB

3. Una vez creados los dos archivos nuevos, abrimos el Docker-compose.yaml. el cual por determinado se abre en VisualStudio, también se puede abrir por bloc de notas.



4. En el archivo Docker-compose.yml haremos la estructura yml para crear un contenedor con volumen y la base de datos.

```
docker-compose.yml X
C:\Users\Lenovo\Desktop> Supermercado > docker-compose.yml
1
2 version: '3.8'
3
4 services:
5   mysql:
6     image: mysql:latest
7     container_name: Supermercado
8     restart: always
9     environment:
10      MYSQL_ROOT_PASSWORD: "1234"
11      MYSQL_DATABASE: "Super"
12      MYSQL_USER: "user"
13      MYSQL_PASSWORD: "password"
14
15     ports:
16       - "3307:3306"
17
18     volumes:
19       - "/c:/Users/Lenovo/Desktop/Supermercado/Supermercado.yml/Taller:/var/lib/mysql"
20       - "/init.sql:/docker-entrypoint-initdb.d/init.sql"
21
22
23
24
25
```

5. Como siguiente se abre el archivo init.sql donde vamos a hacer la tablas de la base de datos e ingresamos los registros para cada tabla, en este caso lo haremos 3 registros.

```
init.sql X
C:\Users\Lenovo\Desktop> Supermercado > init.sql
1 CREATE TABLE Productos (
2   id_producto INT AUTO_INCREMENT PRIMARY KEY,
3   nombre VARCHAR(100) NOT NULL,
4   descripcion TEXT,
5   precio DECIMAL(10,2) NOT NULL,
6   stock INT NOT NULL
7 );
8
9 CREATE TABLE Clientes (
10  id_cliente INT AUTO_INCREMENT PRIMARY KEY,
11  nombre VARCHAR(100) NOT NULL,
12  correo VARCHAR(100) UNIQUE,
13  telefono VARCHAR(15),
14  direccion TEXT
15 );
16
17 CREATE TABLE Ventas (
18  id_venta INT AUTO_INCREMENT PRIMARY KEY,
19  id_cliente INT,
20  fecha_venta DATETIME DEFAULT CURRENT_TIMESTAMP,
21  total DECIMAL(10,2) NOT NULL,
22  FOREIGN KEY (id_cliente) REFERENCES Clientes(id_cliente) ON DELETE SET NULL
23 );
24
```

```
25 CREATE TABLE Detalle_Venta (
26  id_detalle INT AUTO_INCREMENT PRIMARY KEY,
27  id_venta INT,
28  id_producto INT,
29  cantidad INT NOT NULL,
30  precio_unitario DECIMAL(10,2) NOT NULL,
31  subtotal DECIMAL(10,2) NOT NULL,
32  FOREIGN KEY (id_venta) REFERENCES Ventas(id_venta) ON DELETE CASCADE,
33  FOREIGN KEY (id_producto) REFERENCES Productos(id_producto) ON DELETE CASCADE
34 );
35
36 INSERT INTO Clientes (nombre, correo, telefono, direccion) VALUES
37 ('Juan Perez', 'juan@gmail.com', '3001234567', 'Calle 123, Bogota'),
38 ('Maria Gomez', 'maria@gmail.com', '3100876543', 'Carrera 45, Medellin'),
39 ('Carlos Rodriguez', 'carlos@gmail.com', '3205678901', 'Avenida Siempre Viva, Cali');
40
41 INSERT INTO Productos (nombre, descripcion, precio, stock) VALUES
42 ('Arroz', 'Paquete de 1kg', 5.500, 100),
43 ('Leche', 'Botella de 1 litro', 3.750, 50),
44 ('Pan', 'Paquete de 6 unidades', 2.250, 80);
45
```

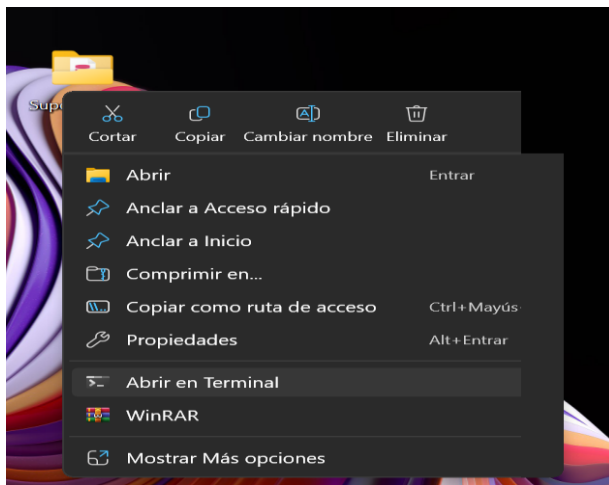
```

26 INSERT INTO Clientes (nombre, correo, telefono, direccion) VALUES
27 ('Juan Perez', 'juan@gmail.com', '3001234567', 'Calle 123, Bogota'),
28 ('Maria Gomez', 'maria@gmail.com', '3109876543', 'Carrera 45, Medellin'),
29 ('Carlos Rodriguez', 'carlos@gmail.com', '3205678901', 'Avenida Siempre Viva, Cali');
30
31 INSERT INTO Productos (nombre, descripcion, precio, stock) VALUES
32 ('Arroz', 'Paquete de 1kg', 5.500, 100),
33 ('Leche', 'Botella de 1 litro', 3.750, 50),
34 ('Pan', 'Paquete de 6 unidades', 2.250, 80);
35
36 INSERT INTO Ventas (id_cliente, total) VALUES
37 (1, 11.000),
38 (2, 7.500),
39 (3, 16.705);
40
41 INSERT INTO Detalle_Venta (id_venta, id_producto, cantidad, precio_unitario, subtotal) VALUES
42 (1, 1, 2, 5.500, 11.000),
43 (2, 2, 2, 3.750, 7.500),
44 (3, 3, 5, 2.250, 11.250);
45
46

```

- El siguiente paso vamos a el contenedor que acabamos de crear el archivo Docker-compose.yml y a su vez levantar la base de datos. Esto lo haremos con el comando Docker-compose up -d en la terminal.

Primero los vamos a para en la carpeta donde tenemos los dos archivos.



```

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\Lenovo\Desktop\Supermercado>

```

el siguiente paso es que vamos a levantar el contenedor con el comando Docker-compose up -d

```

PS C:\Users\Lenovo\Desktop\Supermercado> docker-compose up -d
time="2025-03-20T20:54:27-05:00" level=warning msg="C:\\Users\\Lenovo\\Desktop\\Supermercado\\docker-compose.yml: 'version' is obsolete"
[+] Running 2/2
 ✓ Network supermercado_default Created 0.1s
 ✓ Container Supermercado Started 1.0s
PS C:\Users\Lenovo\Desktop\Supermercado>

```

Como sabemos si está bien abrimos el Docker desktop

<input type="checkbox"/>	▼	●	-	supermercado	-	-	1.43%	1	<input type="checkbox"/>	⋮	🗑
<input type="checkbox"/>		●	3307:3306 ↗	Supermecado	b5391ad1e45c	mysql:lates	1.43%	1	<input type="checkbox"/>	⋮	🗑

Quedo bien levantado el contenedor, nos está respondiendo con un nombre, los puertos por donde está corriendo y lo más importante es que no está pausando la ejecución.

7. Ahora nos vamos a dirigir a la carpeta donde tenemos el proyecto.

📁	Supermercado.yml	20/03/2025 8:54 p.m.	Carpeta de archivos	
📄	docker-compose.yml	20/03/2025 1:08 p.m.	Archivo de origen Ya...	1 KB
📄	init.sql	20/03/2025 2:17 p.m.	Archivo de origen SQL	2 KB

Podemos observar que el contenedor se le levanta correctamente.

8. Vamos a revisar que se creó el volumen

Nombre	Fecha de modificación	Tipo	Tamaño
📁 #innodb_redo	20/03/2025 8:55 p.m.	Carpeta de archivos	
📁 #innodb_temp	20/03/2025 8:55 p.m.	Carpeta de archivos	
📁 mysql	20/03/2025 8:54 p.m.	Carpeta de archivos	
📁 performance_schema	20/03/2025 8:54 p.m.	Carpeta de archivos	
📁 Super	20/03/2025 8:55 p.m.	Carpeta de archivos	
📁 sys	20/03/2025 8:54 p.m.	Carpeta de archivos	
📄 #ib_16384.0.dblwr	20/03/2025 8:55 p.m.	Archivo DBLWR	6.144 KB
📄 #ib_16384.1.dblwr	20/03/2025 8:54 p.m.	Archivo DBLWR	14.336 KB
📄 auto.cnf	20/03/2025 8:54 p.m.	Archivo CNF	1 KB
📄 binlog.000001	20/03/2025 8:55 p.m.	Archivo 000001	2.879 KB
📄 binlog.000002	20/03/2025 8:55 p.m.	Archivo 000002	1 KB
📄 binlog.index	20/03/2025 8:55 p.m.	Archivo INDEX	1 KB
📄 ca.pem	20/03/2025 8:54 p.m.	Archivo PEM	2 KB
📄 ca-key.pem	20/03/2025 8:54 p.m.	Archivo PEM	2 KB
📄 client-cert.pem	20/03/2025 8:54 p.m.	Archivo PEM	2 KB
📄 client-key.pem	20/03/2025 8:54 p.m.	Archivo PEM	2 KB
📄 ib_buffer_pool	20/03/2025 8:55 p.m.	Archivo	6 KB

9. Vamos a revisar si en la carpeta que tiene el nombre de la base de datos se guardaron las tablas que creamos en el init.sql

📁	Super	20/03/2025 8:55 p.m.	Carpeta de archivos	
📄	clientes.ibd	20/03/2025 8:55 p.m.	Archivo IBD	128 KB
📄	detalle_venta.ibd	20/03/2025 8:55 p.m.	Archivo IBD	144 KB
📄	productos.ibd	20/03/2025 8:55 p.m.	Archivo IBD	112 KB
📄	ventas.ibd	20/03/2025 8:55 p.m.	Archivo IBD	128 KB

10. Y ya como ultimo paso abrimos el Mysql Workbench. Donde haremos la conexión con la base de datos y también verificamos si los registros quedaron guardados en la base de datos.

The screenshot displays the MySQL Workbench interface. On the left, a connection window for 'Super' is shown with the user 'root' and host '127.0.0.1:3307'. The main workspace is divided into several panes. The 'Navigator' pane on the right shows the 'Super' database schema with tables: Clientes, Detalle_Venta, Productos, and Ventas. Below the Navigator, four query windows are open, each displaying the results of a 'SELECT * FROM' query. The first query window shows the 'Clientes' table with columns: id_cliente, nombre, correo, telefono, direccion. The second query window shows the 'Detalle_Venta' table with columns: id_detalle, id_venta, id_producto, cantidad, precio_unitario, subtotal. The third query window shows the 'Productos' table with columns: id_producto, nombre, descripcion, precio, stock. The fourth query window shows the 'Ventas' table with columns: id_venta, id_cliente, fecha_venta, total. Each query window has a 'Result Grid' tab and a 'Filter Rows' field.

MySQL Workbench

Super - Warning - not support...

File Edit View Query Database S

Navigator

SCHEMAS

Filter objects

Super

Tables

- Clientes
- Detalle_Venta
- Productos
- Ventas

1 • SELECT * FROM Super.Clientes;

1 • SELECT * FROM Super.Detalle_Venta;

1 • SELECT * FROM Super.Productos;

1 • SELECT * FROM Super.Ventas;

id_cliente	nombre	correo	telefono	direccion
1	Juan Perez	juan@gmail.com	3001234567	Calle 123, Bogota
2	Maria Gomez	maria@gmail.com	3109876543	Carrera 45, Medellin
3	Carlos Rodriguez	carlos@gmail.com	3205678901	Avenida Siempre Viva, Cali
NULL	NULL	NULL	NULL	NULL

id_detalle	id_venta	id_producto	cantidad	precio_unitario	subtotal
1	1	1	2	5.50	11.00
2	2	2	2	3.75	7.50
3	3	3	5	2.25	11.25
NULL	NULL	NULL	NULL	NULL	NULL

id_producto	nombre	descripcion	precio	stock
1	Arroz	Paquete de 1kg	5.50	100
2	Leche	Botella de 1 litro	3.75	50
3	Pan	Paquete de 6 unidades	2.25	80
NULL	NULL	NULL	NULL	NULL

id_venta	id_cliente	fecha_venta	total
1	1	2025-03-21 01:55:02	11.00
2	2	2025-03-21 01:55:02	7.50
3	3	2025-03-21 01:55:02	16.71
NULL	NULL	NULL	NULL

Referencias

- JSON. (s.f.). Introducing JSON. JSON.org. Recuperado el 20 de marzo de 2025, de <https://www.json.org/json-en.html>
- YAML. (s.f.). YAML Ain't Markup Language (YAML). yaml.org. Recuperado el 20 de marzo de 2025, de <https://yaml.org>
- Docker Inc. (s.f.). Overview of Docker Compose. Docker Docs. Recuperado el 20 de marzo de 2025, de <https://docs.docker.com/compose/>
- Oracle Corporation. (s.f.). MySQL 8.0 Reference Manual. MySQL Developer Zone. Recuperado el 20 de marzo de 2025, de <https://dev.mysql.com/doc/refman/8.0/en/>
- Docker Inc. (s.f.). Use MySQL with Docker Compose. Docker Docs. Recuperado el 20 de marzo de 2025, de <https://docs.docker.com/samples/mysql/>
- YAML frente a JSON: diferencia entre los formatos de serialización de datos - AWS. (n.d.). Amazon Web Services, Inc. <https://aws.amazon.com/es/compare/the-difference-between-yaml-and-json/>