

TALLER PRACTICO - CONSULTAS

Manuel Santiago Malpica Daza

ID: 833776

Ingeniería de Sistemas

Semestre 7

Arquitectura

NRC-66168

UNIMINUTO

Corporación Universitaria Minuto de Dios

Docente:

William Alexander Matallana Porras

1. Diferencias entre Archivos. yml vs json

Característica	YAML (.yaml)	JSON (.json)
Sintaxis	Más legible para humanos, usa indentación y no requiere comillas ni llaves.	Usa llaves {} y corchetes [], más estructurado y estricto.
Facilidad de lectura	Fácil de leer y escribir debido a su estructura limpia y minimalista.	Menos legible para humanos debido a su formato más denso.
Comentarios	Permite comentarios con #.	No admite comentarios.
Uso de tipos de datos	Soporta tipos como listas, mapas, booleanos, números y cadenas sin comillas.	Soporta los mismos tipos, pero las cadenas deben estar entre comillas.
Popularidad	Usado en configuración de aplicaciones, Kubernetes, Ansible, CI/CD.	Común en APIs, bases de datos NoSQL (MongoDB), intercambio de datos.
Compatibilidad con lenguajes	Menos soporte nativo, pero existen bibliotecas para la mayoría de lenguajes.	Soporte nativo en casi todos los lenguajes modernos.
Tamaño	Más compacto en configuraciones debido a su sintaxis simplificada.	Puede ser más grande debido a la necesidad de llaves y comillas.

2. Docker – compose .yaml – uso

Docker Compose es una herramienta flexible que te permite definir y manejar aplicaciones de múltiples contenedores de manera simple. Con Docker Compose, puedes definir la configuración de tu entorno de desarrollo en un archivo YAML, indicando los servicios, volúmenes y redes requeridos para tu aplicación. Después, con un único comando, puedes generar y ejecutar todos los contenedores que están definidos en tu archivo de configuración.

3. Como se crea un contenedor usando yaml

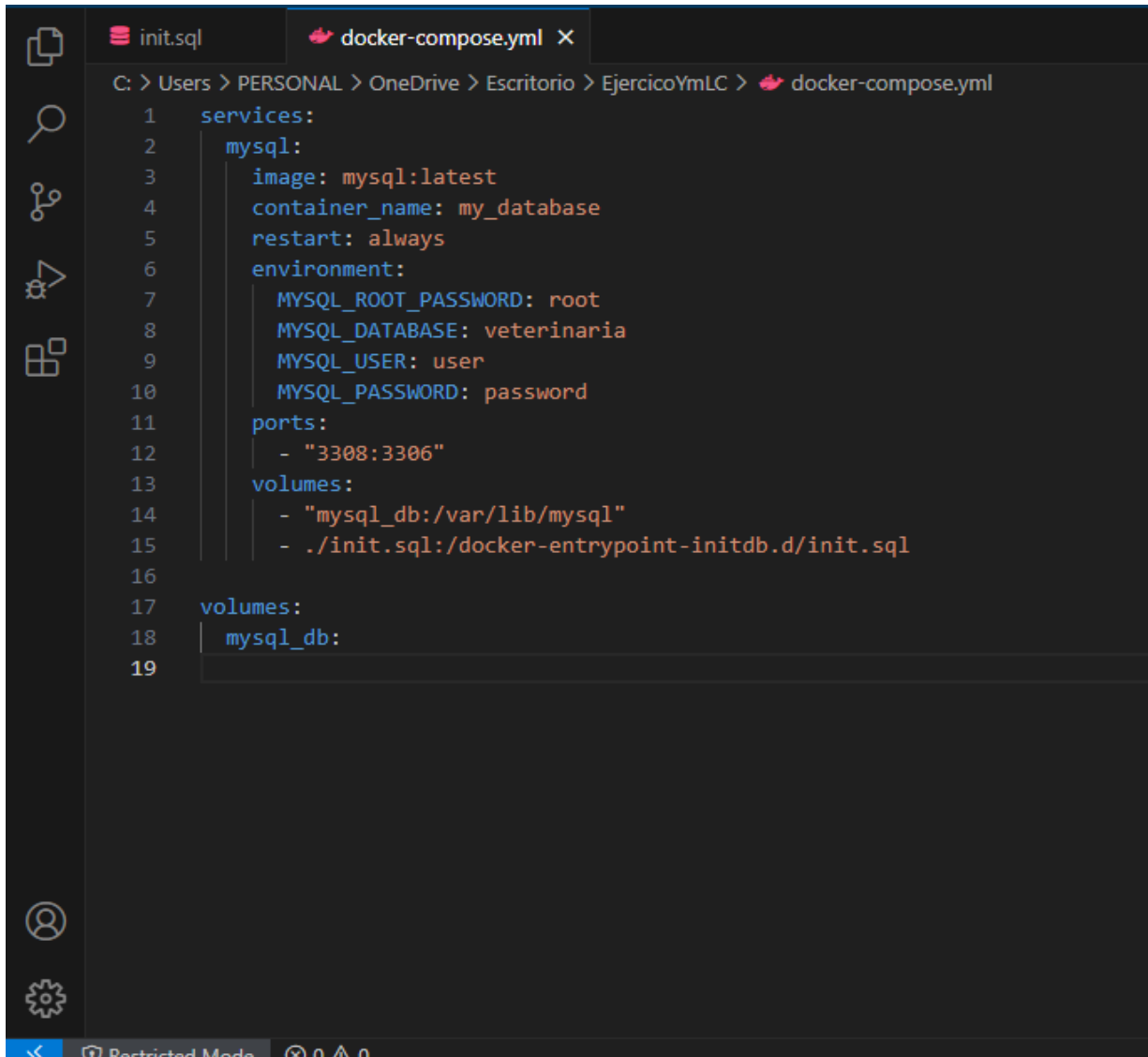
Para crear un contenedor con docker-compose.yaml, primero se debe crear un directorio donde estará el archivo de configuración. Luego, dentro de este directorio, se crea un archivo llamado docker-compose.yaml, donde se define la estructura del contenedor, especificando la imagen, los puertos, las redes y otros parámetros. Por ejemplo, para desplegar un servidor Nginx, el archivo contendría una sección services con un servicio llamado web, que usa la imagen nginx:latest y mapea el puerto 8080 del host al 80 del contenedor. Finalmente, desde la terminal y dentro del mismo directorio, se ejecuta docker-compose up -d para iniciar el contenedor en segundo plano.

En el fichero **docker-compose.yaml**, vamos a definir el escenario de ejecución. El programa **Docker Compose** debe ejecutarse en el mismo directorio donde se encuentra este fichero.

Por lo tanto, cada aplicación que queramos desplegar tendrá su propio directorio con un fichero **docker-compose.yaml** específico.

Por ejemplo, para desplegar la aplicación **Let's Chat**, podríamos tener un directorio con el siguiente contenido en su archivo **docker-compose.yaml**:

Paso 1: Crear el archivo docker-compose.yml

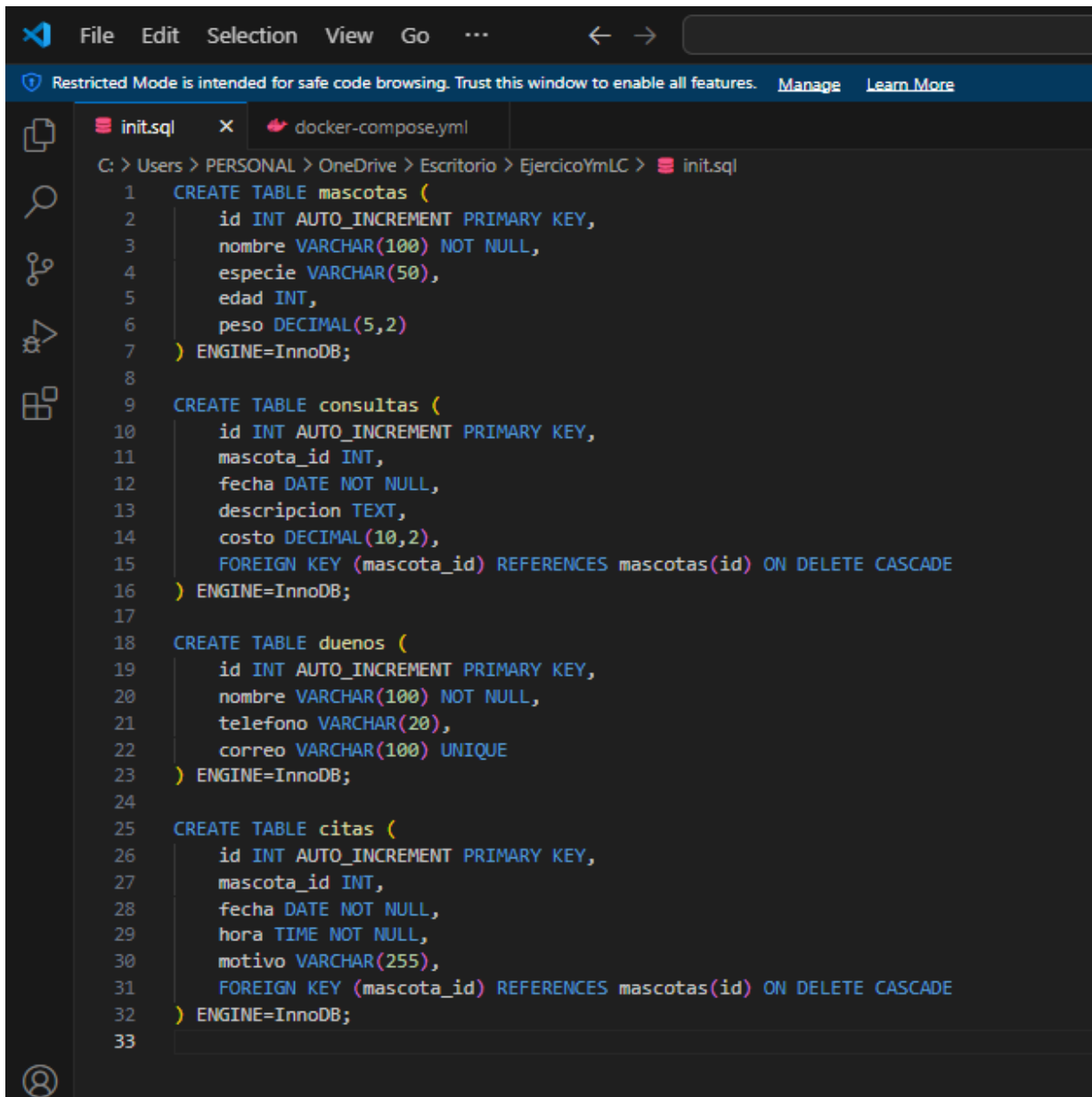


```
1  services:
2    mysql:
3      image: mysql:latest
4      container_name: my_database
5      restart: always
6      environment:
7        MYSQL_ROOT_PASSWORD: root
8        MYSQL_DATABASE: veterinaria
9        MYSQL_USER: user
10       MYSQL_PASSWORD: password
11   ports:
12     - "3308:3306"
13   volumes:
14     - "mysql_db:/var/lib/mysql"
15     - ./init.sql:/docker-entrypoint-initdb.d/init.sql
16
17 volumes:
18   mysql_db:
```

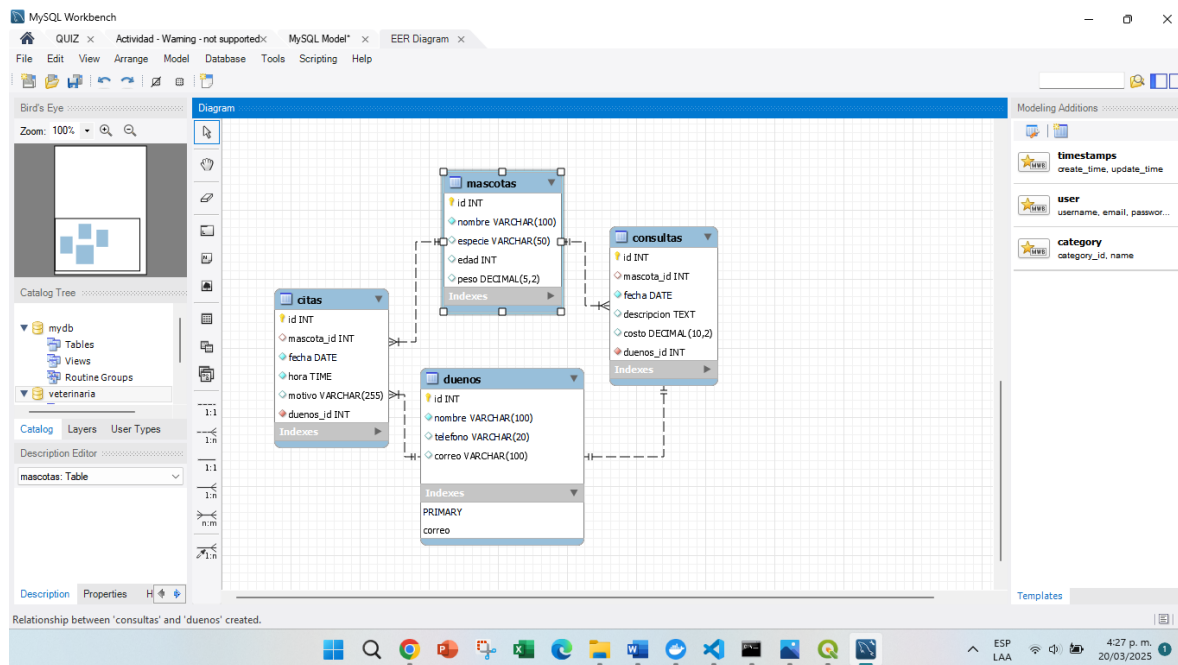
- **services:** Define los servicios que se van a levantar en Docker.
- **mysql:** Es el nombre del servicio que usa la imagen oficial de MySQL.
- **container_name: my_database:** Nombra el contenedor para que se pueda referenciar fácilmente.
- **ports: "3308:3306":** Mapea el puerto 3306 (interno de MySQL) al 3308 en tu máquina.
- **volumes:** Guarda los datos para que no se pierdan al reiniciar el contenedor.

2.Crear el Archivo init.sql

Este archivo contiene las instrucciones para crear las tablas en MySQL se definen 4 tablas (mascotas, consultas, dueños, citas, Y se Agrega claves primarias (id) y relaciones (FOREIGN KEY)).

A screenshot of a code editor window. The title bar shows 'File Edit Selection View Go ...' and navigation arrows. Below the title bar, a status bar reads 'Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More'. The editor has two tabs: 'init.sql' (active) and 'docker-compose.yml'. The file path is 'C: > Users > PERSONAL > OneDrive > Escritorio > EjercicioYmLC > init.sql'. The code defines four MySQL tables: 'mascotas', 'consultas', 'dueños', and 'citas'. Each table has an 'id' as a primary key. 'consultas' and 'citas' have a foreign key 'mascota_id' that references 'mascotas(id)' with 'ON DELETE CASCADE'. The code is as follows:

```
1 CREATE TABLE mascotas (  
2     id INT AUTO_INCREMENT PRIMARY KEY,  
3     nombre VARCHAR(100) NOT NULL,  
4     especie VARCHAR(50),  
5     edad INT,  
6     peso DECIMAL(5,2)  
7 ) ENGINE=InnoDB;  
8  
9 CREATE TABLE consultas (  
10    id INT AUTO_INCREMENT PRIMARY KEY,  
11    mascota_id INT,  
12    fecha DATE NOT NULL,  
13    descripcion TEXT,  
14    costo DECIMAL(10,2),  
15    FOREIGN KEY (mascota_id) REFERENCES mascotas(id) ON DELETE CASCADE  
16 ) ENGINE=InnoDB;  
17  
18 CREATE TABLE dueños (  
19    id INT AUTO_INCREMENT PRIMARY KEY,  
20    nombre VARCHAR(100) NOT NULL,  
21    telefono VARCHAR(20),  
22    correo VARCHAR(100) UNIQUE  
23 ) ENGINE=InnoDB;  
24  
25 CREATE TABLE citas (  
26    id INT AUTO_INCREMENT PRIMARY KEY,  
27    mascota_id INT,  
28    fecha DATE NOT NULL,  
29    hora TIME NOT NULL,  
30    motivo VARCHAR(255),  
31    FOREIGN KEY (mascota_id) REFERENCES mascotas(id) ON DELETE CASCADE  
32 ) ENGINE=InnoDB;  
33
```



Lo que hice fue configurar y desplegar un contenedor de MySQL utilizando Docker Compose en Windows. Primero, creé una carpeta de trabajo llamada **EjercicioYmLC**, donde guardé los archivos necesarios para la configuración.

Dentro de esta carpeta, generé el archivo `docker-compose.yml`, que define los servicios de Docker, y el archivo `init.sql`, que contiene las instrucciones para la creación de tablas en la base de datos.

En el archivo `docker-compose.yml`, especificué la imagen de MySQL que iba a usar, configuré las variables de entorno como la contraseña de root y el nombre de la base de datos, y definí los volúmenes para que los datos se almacenaran de manera persistente. También vinculé un script de inicialización (`init.sql`) para que se ejecutara al iniciar el contenedor.

En este script escribí las sentencias SQL necesarias para crear cuatro tablas: **mascotas**, **consultas**, **dueños** y **citas**, asegurando que la base de datos veterinaria tuviera una estructura adecuada desde el inicio.

Se crea el contenedor y ejecutara automáticamente el script `init.sql`. Para verificar que todo estuviera funcionando correctamente.

Referencias

- <https://www.ibm.com/mx-es/topics/yaml#:~:text=JSON%20suele%20preferirse%20por%20su,tipos%20de%20documentaci%C3%B3n%20e%20intercambio>
- <https://imaginaformacion.com/tutoriales/que-es-docker-compose>