PRÁCTICA DE EVALUACIÓN NIVEL 1

Para afianzar conocimiento y con el fin, de poder pasar a siguientes secciones dejando la parte de dockers finalizada, vamos a realizar una práctica sencilla pero laboriosa, que albergará prácticamente todos los aspectos que hemos ido viendo durante las clases, algunos, tendréis que investigarlos un poco, os pido que salvo que os perdáis completamente no hagáis uso de ChatGPT puesto os dará la solución más directa, pero no adquiriremos los conceptos que se buscan con esta práctica.

Objetivos de la práctica:

- Demostrar la habilidad para crear y gestionar imágenes y contenedores con Docker.
- Entender y aplicar el uso de volúmenes para la persistencia de datos.
- Configurar redes personalizadas para la comunicación entre contenedores.
- Utilizar Docker Compose para gestionar aplicaciones multi-contenedor.

Parte 1: Dockerfile y Creación de Imágenes (30 minutos)

Tarea: Crear un Dockerfile para una aplicación web simple.
Especificaciones:

- **Base:** Utiliza node:14-alpine como imagen base.
- **Trabajo:** Establece un directorio de trabajo /app.
- **Dependencias**: Copia un archivo package.json y ejecuta npm install.
- **Código fuente**: Copia el resto del código fuente.
- **Puerto:** Expone el puerto 3000.
- **Comando:** Usa npm start para iniciar la aplicación.

Preguntas de seguimiento:

1. ¿Qué hace cada instrucción en el Dockerfile?
2. ¿Cómo construirías esta imagen?
3. ¿Cómo ejecutarías un contenedor basado en esta imagen?

Parte 2: Docker Compose y Servicios Multi-Contenedor (30 minutos)

Tarea: Definir un **docker-compose.yml** para una aplicación que incluya los servicios de la aplicación web y una base de datos Redis.

Especificaciones:

- **Servicio Web:**
  - Usa la imagen construida en la Parte 1.
  - Mapea el puerto 3000 al puerto 3000 del host.
  - Conecta a una red llamada **app-network**.
- **Servicio Redis:**
  - Utiliza la imagen **redis:alpine**.
  - No necesita mapeo de puertos.
  - Conecta a la misma red **app-network**.
- **Redes:**
  - Define **app-network** como una red de tipo bridge.

Preguntas de seguimiento:

1. Explica el propósito de cada servicio en el archivo **docker-compose.yml**.
2. ¿Cómo levantarías todos los servicios con Docker Compose?
3. ¿Cómo detendrías todos los servicios y eliminarías los recursos creados?

Parte 3: Volumen y Persistencia de Datos (20 minutos)

Tarea: Modificar el **docker-compose.yml** para incluir un volumen que persista los datos de Redis.

Especificaciones:

- **Volumen para Redis:**
  - Define un volumen llamado redis-data para persistir los datos de Redis.
  - Monta este volumen en /data dentro del contenedor de Redis.

Preguntas de seguimiento:

1. ¿Por qué es importante la persistencia de datos para un servicio como Redis?
2. Describe cómo configurarías un volumen para otro tipo de servicio, como una base de datos SQL.

Parte 4: Redes en Docker (20 minutos)

Tarea: Explicar y configurar una red personalizada para optimizar la comunicación entre contenedores.

Especificaciones:

- **Tipo de Red:**
  - Crea una red de tipo bridge llamada app-network.
- **Conectividad:**
  - Asegura que el servicio web y Redis solo puedan comunicarse entre ellos y no con el exterior directamente, excepto que el servicio web debe ser accesible desde el host.

Preguntas de seguimiento:

1. ¿Cuáles son los beneficios de utilizar redes personalizadas en Docker?
2. Explica cómo el aislamiento de red puede mejorar la seguridad en aplicaciones Dockerizadas.

Criterios de Evaluación

• **Correctitud del Código:** Se verificarán que los Dockerfiles y docker-compose.yml sean sintácticamente correctos y funcionales.
• **Comprensión Conceptual:** Se valorarán las respuestas a preguntas de seguimiento para medir la comprensión de Docker.
• **Práctica de Comandos:** Se revisará la habilidad para ejecutar comandos relevantes de Docker y Docker Compose.