Asignatura	Datos del alumno	Fecha
Contenedores	Apellidos: Santana de la rosa	11/12/2024
	Nombre: Miguel Angel	

Actividad. Creación de aplicativos en contendores con diferentes lenguajes de programación

- **Objetivos** de la actividad:

Desarrollar los conocimientos obtenidos a través de los temas dedicados a creación de Docker en la asignatura de Contenedores.

- **Descripción** de la actividad y **pautas** de elaboración:

Hay diferentes lenguajes de programación para la creación de Docker. Los más utilizados actualmente son Node.js, Python, Java, Go, C# y Rust.

En esta actividad te pedimos que construyas y ejecutes tres contenedores en cuatro lenguajes de programación diferentes de los anteriormente citados. Puedes utilizar código que encuentres por internet, por supuesto que no hay que escribir ninguna línea de código, sino que hay que construir los contenedores con aplicaciones que puedes encontrar en repositorios Git.

Después de crearlos ejecuta los Dockers cada uno por un puerto distinto. También deberás de subir las imágenes a un repositorio de Dockers, puedes usar Docker Hub o, también, puedes utilizar otros repositorios, en el caso de que conozcas otros por temas profesionales. Deberás documentar los pasos con uno de los Dockers construidos.

Por otro lado, se buscará uns aplicación que esté creada con contenedores y que tenga también una base de datos en contenedor y se tendrá que ejecutar con Docker Compose.

Se deberá de documentar los pasos completamente y explicar los ficheros Dockerfile y Docker-compose.yaml.

- **Extensión** máxima de la actividad:

Asignatura	Datos del alumno	Fecha
Contenedores	Apellidos: Santana de la rosa	11/12/2024
	Nombre: Miguel Angel	

La entrega consistirá en un archivo ZIP con los ficheros de los ejemplos de los cuatro Dockers construidos y un informe en PDF de máximo quince páginas (sin contar el código incluido en el ZIP). **Formato** del PDF fuente de texto Calibri, tamaño 11 e interlineado 1,5.

Rúbrica

Ver Excel adjunto.

Descripción del archivo docker-compose.yml

El archivo que has proporcionado configura un entorno multi-contenedor utilizando **Docker Compose**. Este entorno incluye varias aplicaciones (Node.js, .NET, Python, y Rust) que dependen de una base de datos MySQL común. A continuación, te guiaré paso a paso como un profesor de DevOps, explicando cada sección y su propósito.

Asignatura	Datos del alumno	Fecha
Contenedores	Apellidos: Santana de la rosa	11/12/2024
	Nombre: Miguel Angel	

Respuesta a Practica de Docker.

1. Servicios Definidos

Cada servicio en Docker Compose representa un contenedor. En este caso, tienes cinco servicios: cuatro aplicaciones (Node.js, .NET, Python, y Rust) y un contenedor para la base de datos MySQL.

a) Servicio nodejs-app

- **Contexto de construcción:** La aplicación Node.js se encuentra en el directorio ./nodejs-app y utiliza un Dockerfile para construir la imagen.
- **Puertos:** Se redirige el puerto 4000 del host al puerto 4000 del contenedor.
- **Variables de entorno:** Configuran las credenciales y detalles de conexión para que Node.js se comunique con MySQL.
- **Dependencias:** Declara que este contenedor depende del servicio mysql, asegurando que MySQL esté listo antes de iniciar.
- **Volúmenes:** Sincroniza el código fuente local (./nodejs-app) con el directorio /app del contenedor.
- **Comando:** Inicia la aplicación con npm start.

b) Servicio dotnet-app

- Similar al servicio anterior, pero:
    - Su código fuente está en ./dotnet-app/src.
    - Expone el puerto 7001.
    - Ejecuta la aplicación .NET utilizando el comando dotnet app.dll.

c) Servicio python-app

- Configuración para una aplicación Python:
    - Su código fuente está en ./python-app.

- Usa python app.py como comando para iniciar.
- Expone el puerto 8000.

d) Servicio rust-app

- Configuración para una aplicación Rust:
  - El código fuente está en ./rust-app.
  - Utiliza cargo run para compilar y ejecutar la aplicación.
  - Expone el puerto 9000.

2. Servicio mysql (Base de Datos)

Este servicio configura un contenedor con MySQL:

- **Imagen:** Usa la imagen oficial de MySQL.
- **Puertos:** Expone el puerto 3306 para que las aplicaciones puedan conectarse desde el host.
- **Variables de entorno:** Configura las credenciales para la base de datos:
  - MYSQL_ROOT_PASSWORD: Contraseña del usuario root.
  - MYSQL_DATABASE: Nombre de la base de datos inicial.
  - MYSQL_USER: Usuario MySQL adicional.
  - MYSQL_PASSWORD: Contraseña del usuario adicional.
- **Volúmenes:** Monta un volumen persistente (./mysql-data) para guardar los datos de la base de datos, incluso si el contenedor es eliminado.

3. Volúmenes

El volumen mysql-data utiliza el controlador local para almacenar los datos de MySQL de forma persistente. Esto asegura que los datos no se pierdan al reiniciar el contenedor.

Asignatura	Datos del alumno	Fecha
Contenedores	Apellidos: Santana de la rosa	11/12/2024
	Nombre: Miguel Angel	

4. Flujo de Trabajo de DevOps

Paso 1: Preparación

- Asegúrate de que Docker y Docker Compose estén instalados en tu máquina.
- Estructura los directorios como se indica (cada aplicación debe tener su propia carpeta con un Dockerfile y el código fuente).

Paso 2: Construcción y Ejecución

- Ejecuta docker-compose up --build en el directorio donde está este archivo.
    - **--build:** Fuerza la construcción de las imágenes antes de iniciar los contenedores.
- Docker Compose creará la red y gestionará las dependencias automáticamente.

Paso 3: Monitoreo

- Usa docker ps para verificar que todos los contenedores están corriendo.
- Los logs pueden revisarse con docker-compose logs -f.

Paso 4: Acceso

- Cada aplicación está disponible en el puerto correspondiente:
    - Node.js: http://localhost:4000
    - .NET: http://localhost:7001
    - Python: http://localhost:8000
    - Rust: http://localhost:9000

Paso 5: Persistencia

- Los datos de MySQL están almacenados en el directorio ./mysql-data en tu sistema local.

Cada imagen debe ser empujada (push) a Docker Hub para que esté disponible públicamente o en repositorios privados.

Asignatura	Datos del alumno	Fecha
Contenedores	Apellidos: Santana de la rosa	11/12/2024
	Nombre: Miguel Angel	

a) Subir la Imagenes de Docker

- Usa el comando docker push para subir las 3 imágenes al repositorio de Docker Hub:

docker push mi_usuario/nodejs:1.0

docker push mi_usuario/dotnet:1.0

docker push mi_usuario/pyhton:1.0

a) Ejecutar la Imagenes de Docker

- Usa el comando docker run para ejecturar las 3 imágenes:

docker run -p 4000:4000 mi_usuario/nodejs:1.0

docker run -p 5001:5001 mi_usuario/dotnet:1.0

docker run -p 8000:8000 mi_usuario/python:1.0