Caso de Estudio - Contenedores

La tienda BOSTON, un minorista de ropa establecido en una ubicación céntrica de la ciudad, busca mejorar su gestión de inventario para optimizar sus operaciones diarias. La tienda se enfrenta a desafíos en el seguimiento preciso de su inventario de productos de ropa, lo que ha resultado en dificultades para mantener el stock adecuado.

Para abordar esta problemática, la dirección de la tienda decide contratar a un estudiante de Ciencia de la Computación para liderar el desarrollo de un proyecto que modernice su sistema de gestión de inventario. En la primera fase del proyecto, se determina implementar una base de datos utilizando PostgreSQL y desarrollar una API con Flask. Además, se establece que la solución será desplegada utilizando contenedores Docker para garantizar la portabilidad y escalabilidad del sistema.

Índice

Docker

- Documentación Docker

1. Despliegue de la Base de Datos

Esquema de la Base de Datos

```
CREATE TABLE IF NOT EXISTS product (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) UNIQUE,
    description TEXT,
    price NUMERIC,
    stock INTEGER
);
```

Tener en cuenta las siguientes variables de entorno para el despliegue:

- `DB_USER` =postgres
- `DB_PASSWORD` =postgres
- `DB_NAME` =BOSTON

Actividad 1.1

Para garantizar la persistencia de los datos registrados, es fundamental contar con un volumen que asegure que dicha información no se pierda al eliminar el contenedor.

- Indicar cuál es el comando para crear un volumen con el nombre “boston-data”:

Pista:

- Documentación Docker - Crear Volumen

Actividad 1.2

- Indicar cuál es el comando para desplegar el contenedor de la base de datos con las siguientes características:
  - Nombrar dicho contenedor como “boston-db”
  - Mapear de forma idéntica con el puerto predeterminado de una base de datos de Postgres (5432)
  - Utilizar la última versión de postgres
  - Asociar dicho contenedor con el volumen creado anteriormente
  - Ejecutar dicho contenedor en segundo plano y debe removerse automáticamente

Pistas

- Documentación Docker - docker run: `docker run --help`
- Asignar variable de entorno: `docker run -e TEST=10`
  - `POSTGRES_USER`, `POSTGRES_PASSWORD`, `POSTGRES_DB`
- Mapeo de puertos del contenedor con el host: `docker run -p <host_port>:<container_port>`
- Ejecutar el contenedor en segundo plano: `docker run -d`
- Remover el contenedor automáticamente: `docker run --rm`
- Asignar el nombre del contenedor: `docker run --name <container_name>`
- La ubicación predeterminada donde PostgreSQL almacena sus datos dentro del contenedor es en: `/var/lib/postgresql/data/`
- Asociar el contenedor con un volumen: `docker run -v <volumen>:<container_path>`

Actividad 1.3

- Indicar cuál es el comando para implementar el esquema de la base de datos.

Pistas:

- Documentación Docker - docker exec: `docker exec --help`
- Utilizar psql para interactuar con las bases de datos de Postgres.

2. Conexión de la API con la Base de Datos

Cuando ejecutamos contenedores de Docker de manera independiente (es decir, sin utilizar docker compose), Docker asigna una dirección IP a cada contenedor que está en la misma red de Docker. Esta dirección IP se utiliza para permitir la comunicación entre los contenedores y con el host en el que se están ejecutando los contenedores.

Docker utiliza un concepto llamado "docker network" para facilitar la comunicación entre contenedores. Por defecto, cuando ejecutas un contenedor sin especificar una red, Docker lo coloca en la red predeterminada llamada bridge. Esta red permite que los contenedores se comuniquen entre sí y con el host utilizando direcciones IP asignadas por Docker.

Para que nuestra aplicación pueda recuperar los datos es necesario conectarnos al contenedor de la base de datos por lo que se requiere determinar la dirección IP de dicho contenedor. Para lo cual utilizaremos el siguiente comando:

```
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' boston-db
```

3. Despliegue de la API

Actividad 3.1

- Diseñe el `Dockerfile` para construir la imagen de la API, con las siguientes características:
  - Utilizar la imagen `python:3.9-slim`.
  - Asignar el directorio de trabajo en `app/`.
  - Indicar el puerto de la aplicación.
  - Copiar los archivos necesarios.

Pista:

- Documentación Docker - Dockerfile

Actividad 3.2

- Indique cuál es el comando para construir la imagen de nuestra aplicación con las siguientes características:
  - El nombre de la imagen es: `app`
  - La etiqueta de la imagen es: `v1.0`

Pistas:

- Documentación Docker - docker build: `docker build --help`
- Asignar una etiqueta a la imagen: `docker run -t`

Actividad 3.3

- Indicar el comando para ejecutar el contenedor de la aplicación con las siguientes características:
  - Mapear el puerto correspondiente
  - Nombrar dicho contenedor como boston-app
  - Ejecutar dicho contenedor en segundo plano y debe removerse automáticamente

4. Docker Hub

Docker Hub es un servicio en la nube proporcionado por Docker que permite a los usuarios almacenar, compartir y administrar imágenes de contenedores Docker de manera centralizada. Es una plataforma integral que facilita la distribución de aplicaciones en forma de contenedores, lo que simplifica la gestión de entornos de desarrollo. Para ello es necesario registrarse en su plataforma.

Actividad 4.1

- Subir la imagen creada a Docker Hub e indicar el enlace público para su acceso.

Pistas:

- Documentación Docker - docker login: `docker login --help`
- Documentación Docker - docker push: `docker push --help`

5. Orquestación de contenedores

Desplegar cada contenedor de forma independiente no es la forma más eficiente, es momento de realizar la orquestación de dichos contenedores a través de `docker compose`.

- Documentación Docker - Compose

Actividad 5.1

- Diseñe el `docker-compose.yml` para poder desplegar los contenedores teniendo en cuenta las consideraciones anteriores.

Actividad 5.2

- Indicar cual es el comando para ejecutar el `docker-compose.yml` en segundo plano.

6. Entregable del Caso de Estudio

- Grabar un video (corto) mostrando los resultados de todas las actividades.
- Tomar como referencia la plantilla `plantilla-solucion.md` para documentar las soluciones de las actividades.