Guía Paso a Paso para Crear y Ejecutar una Aplicación Docker con PostgreSQL y Python ## 1. Construir y Ejecutar los Contenedores ### 1.1. Construir la Imagen Docker para la Aplicación Python ```bash docker-compose build

1.2. Levantar los Contenedores Usando Docker Compose

bash

```
docker-compose up -d
```

1.3. Verificar los Contenedores en Ejecución

bash

```
docker-compose ps
```

1.4. Verificar los Logs del Contenedor de la Aplicación

bash

```
docker logs jsonplaceholder_container
```

1.5. Inspeccionar los Contenedores

bash

```
docker inspect jsonplaceholder_container
```

1.6. Verificar los Volúmenes de Docker

bash

```
docker volume ls
```

1.7. Inspeccionar el Volumen de PostgreSQL

bash

```
docker volume inspect jsonplaceholder_project_postgres_data
```

2. Parar y Eliminar Contenedores

2.1. Parar los Contenedores

bash

```
docker-compose down
```

2.2. Eliminar los Contenedores y Volúmenes

bash

```
docker-compose down --volumes
```

3. Explicación de los Archivos

3.1. Dockerfile

El Dockerfile define cómo construir la imagen Docker para la aplicación Python:

- Usa una imagen base de Python.
- Copia y instala las dependencias.
- Copia el código de la aplicación.
- Define el comando para ejecutar el script Python.

3.2. docker-compose.yml

El `docker-compose.yml` define los servicios necesarios para el proyecto:

- **PostgreSQL**: Base de datos.
- **pgAdmin**: Interfaz gráfica para gestionar la base de datos.
- **json_app**: La aplicación Python que se conecta a la base de datos.

Contenedores

1. Contenedor `jsonplaceholder`

Estado y Detalles del Contenedor

bash

```
docker ps -f name=jsonplaceholder
```

Logs del Contenedor

bash

```
docker logs jsonplaceholder docker logs postgres docker logs imagen
```

Imágenes

2. Imagen `postgres:16`

Listar Imágenes

bash

```
docker images postgres:16
```

Eliminar Imagen

bash

```
docker rmi 159173ed3e65
```

Red

3. Redes Docker

Listar Redes

bash

```
docker network ls
```

Inspeccionar Red

bash

```
docker network inspect bridge
```

Notas adicionales:

- Asegúrate de reemplazar los nombres de contenedores (`jsonplaceholder`, `postgres`, `pgadmin`, `jsonplaceholder_app`) y los IDs de contenedor (`159173ed3e65`, `e0effb7e924e`, `bc9deb53bef9`) con los valores específicos de tu entorno.
- Los comandos están diseñados para administrar los recursos Docker asociados con cada componente del proyecto, incluyendo contenedores, imágenes y redes.
- Si necesitas acciones específicas como detener, iniciar, eliminar o inspeccionar, puedes adaptar estos comandos según tus necesidades y el estado actual de tus servicios Docker.

Contenedores

1. Contenedor `jsonplaceholder`

Estado y Detalles del Contenedor

bash

```
docker ps -f name=jsonplaceholder
```

Logs del Contenedor

bash

```
docker logs jsonplaceholder
```

2. Contenedor `postgres`

Estado y Detalles del Contenedor

bash

```
docker ps -f name=postgres
```

Logs del Contenedor

bash

```
docker logs postgres
```

3. Contenedor `pgadmin`

Estado y Detalles del Contenedor

bash

```
docker ps -f name=pgadmin
```

Logs del Contenedor

bash

```
docker logs pgadmin
```

4. Contenedor `jsonplaceholder_app-1`

Estado y Detalles del Contenedor

bash

```
docker ps -f name=jsonplaceholder_app-1
```

Logs del Contenedor

bash

```
docker logs jsonplaceholder_app-1
```

Imágenes

5. Imagen `postgres:16`

Listar Imágenes

bash

```
docker images postgres:16
```

Eliminar Imagen

bash

```
docker rmi 159173ed3e65
```

6. Imagen `dpage/pgadmin4`

Listar Imágenes

bash

```
docker images dpage/pgadmin4
```

Eliminar Imagen

bash

```
docker rmi e0effb7e924e
```

7. Imagen `jsonplaceholder-jsonplaceholder_app`

Listar Imágenes

bash

```
docker images jsonplaceholder-jsonplaceholder_app
```

Eliminar Imagen

bash

```
docker rmi bc9deb53bef9
```

Red

8. Redes Docker

Listar Redes

bash

```
docker network ls
```

Inspeccionar Red

bash

```
docker network inspect bridge
```

Notas adicionales:

- Asegúrate de reemplazar los nombres de contenedores (`jsonplaceholder`, `postgres`, `pgadmin`, `jsonplaceholder_app-1`) y los IDs de contenedor (`159173ed3e65`, `e0effb7e924e`, `bc9deb53bef9`) con los valores específicos de tu entorno.
- Los comandos proporcionados te permiten administrar y obtener información detallada sobre los contenedores, imágenes y redes Docker asociadas con cada servicio en tu proyecto.
- Puedes ajustar los comandos según sea necesario para realizar acciones específicas como detener, iniciar, eliminar o inspeccionar según el estado y las necesidades actuales de tus servicios Docker.

Comando `docker ps -a`

El comando `docker ps -a` se utiliza para listar todos los contenedores Docker presentes en tu sistema, incluidos los que están actualmente en ejecución y los que han sido detenidos o finalizados. La opción `-a` o `--all` asegura que se muestren todos los contenedores, independientemente de su estado.

Uso común:

1. **Listar todos los contenedores**: Proporciona una visión completa de todos los contenedores Docker en tu sistema, lo que incluye:
    - Contenedores en ejecución: Aquellos que están activos y actualmente procesando tareas.

- Contenedores detenidos: Aquellos que han sido detenidos explícitamente con `docker stop` o que han finalizado naturalmente después de que su proceso principal ha terminado.

2. **Obtener información detallada**:

   - **Nombres y IDs**: Muestra los nombres y los IDs de todos los contenedores, lo que es crucial para realizar operaciones específicas sobre ellos.
   - **Estado actual**: Indica si un contenedor está en ejecución (`Up`) o si ha finalizado (`Exited`).
   - **Último estado**: Información sobre cuándo y cómo finalizó un contenedor.

3. **Gestionar contenedores**: Facilita la gestión de contenedores Docker, permitiendo acciones como:

   - Iniciar contenedores detenidos con `docker start <nombre_o_id>`.
   - Detener contenedores en ejecución con `docker stop <nombre_o_id>`.
   - Eliminar contenedores finalizados con `docker rm <nombre_o_id>`.

Ejemplo de uso:

Para listar todos los contenedores Docker en tu sistema, simplemente ejecuta:

bash

```
docker ps -a
```

Esto te proporcionará una salida similar a la siguiente:

bash

```
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES c0ff33b23a49 nginx:latest "/docker-entrypoint.…" 2 hours ago Up 2 hours 80/tcp web_server abc123def456 postgres:12 "docker-entrypoint.s…" 3 days ago Exited (137) 2 days ago db_server def456ghi789 redis:alpine "docker-entrypoint.s…" 5 days ago Up 5 days
```