Proyecto - Sistemas Operativos

Docker-Compose

Integrantes

Marcelo Alejandro García Millán - 201941427

Kevin Alejandro Vélez Agudelo - 202123281

Asignatura: Sistemas Operativos

Docente: John Alexander Sanabria Ordóñez

Universidad Del Valle

Santiago de Cali

6 de febrero del 2023

Tabla de Contenido

Introducción y Metodología

En este proyecto, se tuvo como finalidad crear una aplicación conformada de dos partes: Una **Interfaz Web** y una **Base de Datos**, el objetivo era gestionar un contador por medio de la codificación de endpoints que manejan métodos de petición HTTP (GET, DELETE, POST Y PUT).

Para la solución de este proyecto se decidieron usar tres componentes clave: **Python** como el entorno de programación de la aplicación web, **Flask** para la creación de la aplicación web, y **PostgreSQL** como el sistema gestor de nuestra Base de Datos, adicionalmente, se usó **Psycopg** como el conector de la aplicación web con la Base de Datos gestionada por Postgres, este es de gran utilidad para hacerle peticiones y consultas a nuestra Base de Datos.

Finalmente, con **Docker** y **Docker Compose** podemos levantar la aplicación con la ayuda de contenedores que se comunicarán de forma conjunta.

Ejecución

Para la ejecución de la aplicación, se requiere de tener como prerrequisito los siguientes componentes en el sistema:

- Docker
- Docker Compose

Una vez con los componentes, con la terminal, nos adentramos en la carpeta del proyecto, y ejecutamos dos comandos:

1. El `docker-compose build`: nos servirá para construir la imagen que contendrá los servicios que especificamos en el archivo `docker-compose.yml`.
2. El `docker-compose up`: nos servirá para construir, iniciar y conectarnos a los contenedores de los servicios y así empezar la ejecución de la aplicación.

Ya con la aplicación en ejecución, podemos empezar a hacer peticiones HTTP, esto lo podemos hacer con el comando `curl -i -X <METHOD> <URL>`, explicado así:

- curl: Es una herramienta de comandos usada para la transferencia de datos hacia una URL especificada.
- -i: Esta bandera nos especifica que queremos que nos imprima en pantalla la salida de la petición HTTP que hemos hecho

- -x <METHOD>: Con esta bandera podemos especificar el método que queremos usar, en <METHOD> iría el método.
- <URL>: Ahí va la URL del sitio web al que queremos hacerle peticiones HTTP.

Dificultades y Soluciones

En el desarrollo del proyecto se presentaron varias dificultades, unas más que otras pero hallamos alternativas o soluciones a ellas. La primera dificultad antes de empezar con el proyecto fue que el subsistema de Linux en el sistema operativo Windows (WSL) tenía muchas incompatibilidades con el programa Docker y uso de muchas librerías para el desarrollo de la aplicación web con Docker-Compose. La alternativa tomada fue instalar un sistema operativo Linux en una máquina virtual, mediante el programa Oracle VM virtualbox. Cada integrante usó una distribución de Linux diferente (Ubuntu y Mint) pero eso no fue impedimento para llevar a cabo el proyecto.

Imagen 1. Distribución de Linux: Ubuntu.

Imagen 2. Distribución de Linux: Mint.

Algo inusual es que para el compañero que usó la distribución de Mint, no tenía permisos de superusuario, se buscó en la web la solución, básicamente se debía crear un grupo con los permisos de superusuario y en él colocar el usuario actualmente logueado.

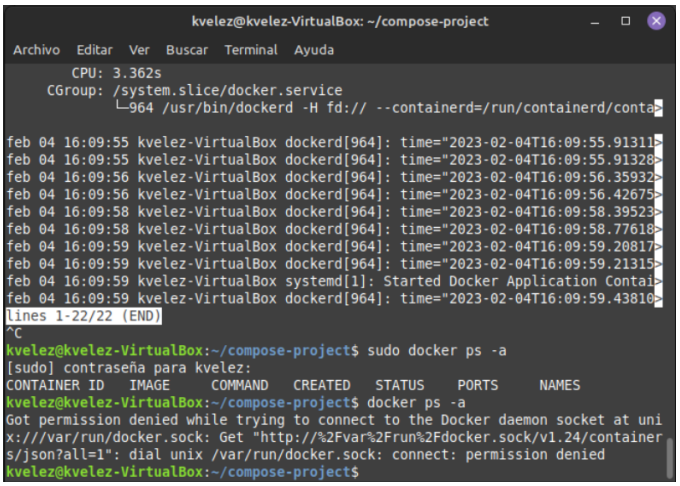

Imagen 3. Comandos en terminal (BASH) sin permisos de superusuario.

Para crear el grupo con permisos administrativos se ejecuta en la terminal el siguiente comando: $ sudo groupadd docker, después, para añadir el usuario se coloca el comando $ sudo usermod -aG docker $USER, luego se loguea en el nuevo grupo creado con: $ newgrp docker, por último se digita un comando de docker para saber si los cambios se hicieron exitosamente, un ejemplo sencillo puede ser consultar las imágenes docker mediante: docker ps -a.

Mientras se iba avanzando en el proyecto, se consultó sobre la API de la base de datos Postgres, se encontró que su puerto era 5432 y que por ende se debía usar otro puerto diferente al que se iba usar para las consultas, usamos el puerto 5555 del anfitrión para la conexión con la base de datos.

```
volumes:
  - ./sql/1schema.sql:/docker-entrypoint-initdb.d/1schema.sql
  - ./sql/2data.sql:/docker-entrypoint-initdb.d/2data.sql
ports:
 - "5555:5432"
```

Imagen 4. Puertos usados para la conexión con API Postgres..

Otra dificultad fue la de lograr la exitosa conexión con la base de datos, se creía que con el puerto era suficiente, investigando nos dimos cuenta que se necesitaba un adaptador de PostgreSQL para la aplicación de Python, este nos permitió hacer las peticiones a la base de datos respectivamente. Se agrega este adaptador en el documento requires.txt.

```
1  flask
2  psycopg2-binary
```

Imagen 5. Documento requires.txt.

Mediante el desarrollo del servidor no tuvimos ninguna complicación, siguiendo las indicaciones del docente y un tutorial alojado en la plataforma de Youtube pudimos completar el servidor con docker-compose. La última dificultad que se presentó fue la ejecución del servidor mediante el comando docker-compose up, la primera vez que lo ejecutamos nos salió un error de que el servidor no existía o que la base de

datos estaba lista para reiniciarse, esto era debido a que primero se creaba el servidor antes que la base de datos.

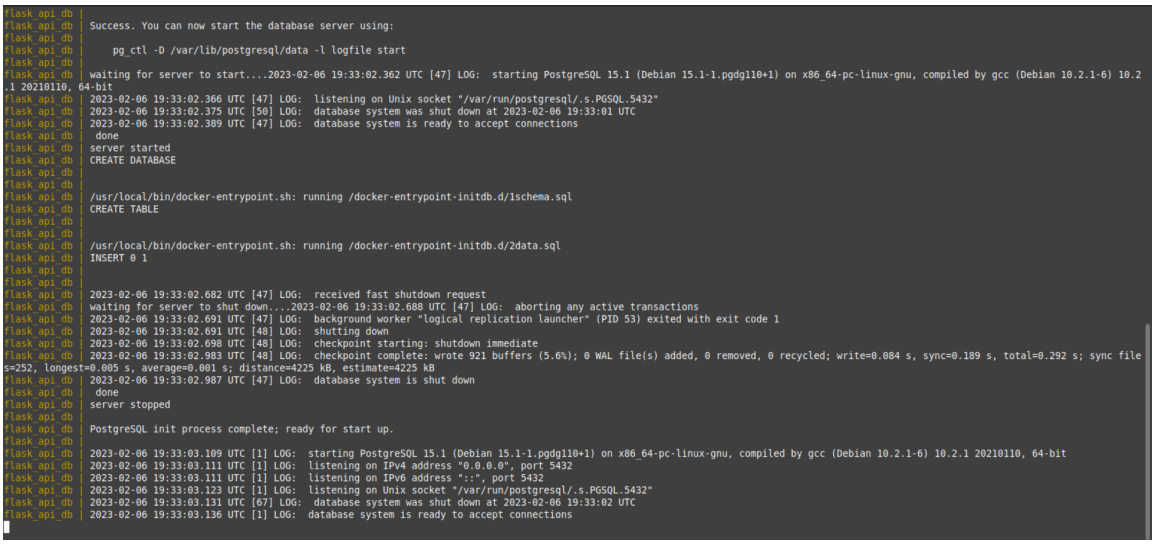

Imagen 6. Foto de terminal Bash dónde se ejecutó el comando: docker-compose up por primera vez.

La solución fue crear una flag (es una condición para el documento docker-compose) dentro del archivo docker-compose.yml.

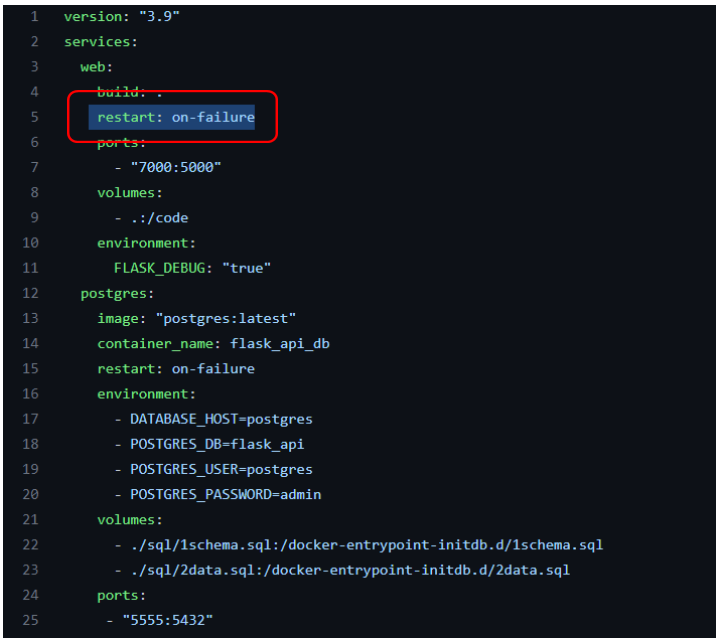

Imagen 8. Documento docker-compose.yml, en la línea 5 se específica el flag colocado.

Este flag nos permite básicamente repetir el proceso de construcción del servidor si sale un error, con esto nos ahorramos volver a ejecutar el comando: docker-compose up.

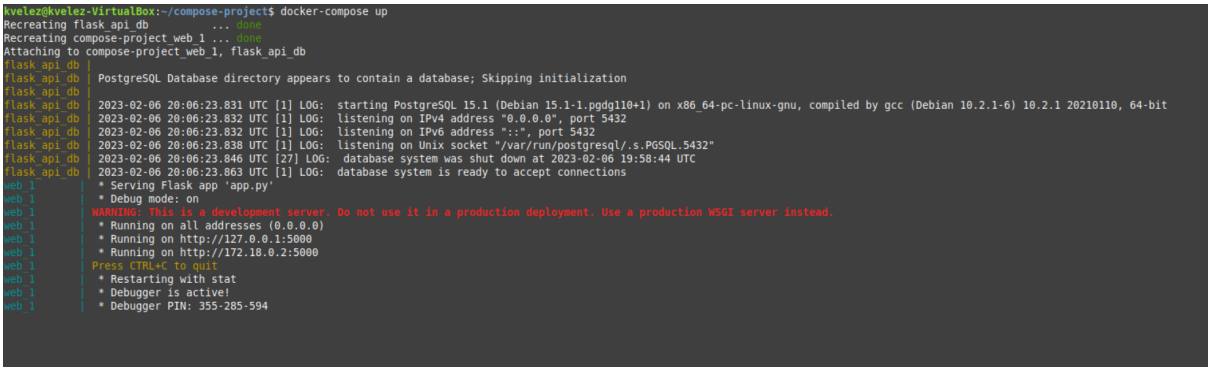

Imagen 9. Docker-compose ejecutándose exitosamente al primer comando de docker-compose up.

Recursos

Repositorio

- https://github.com/MarceloUnivalle/compose-project

Video de la aplicación en YouTube

- https://www.youtube.com/watch?v=5nn-2NprW7w

Referencias

- *The psycopg2 module content — Psycopg 2.9.5 documentation*. (n.d.). Psycopg. Retrieved February 5, 2023, from https://www.psycopg.org/docs/module.html
- *How to fix docker: Got permission denied issue*. (2018, February 23). Stack Overflow. Retrieved February 5, 2023, from https://stackoverflow.com/questions/48957195/how-to-fix-docker-got-permission-denied-issue
- UskoKruM2010. (2022, 23 de abril). *REST API con Python, Flask y PostgreSQL | Crea y Prueba tu REST API (GET, POST, PUT, DELETE)*. [Video]. YouTube. From https://www.youtube.com/watch?v=riijt-xcqYI
- *Compose specification*. (n.d.). Docker Documentation. Retrieved February 6, 2023, from https://docs.docker.com/compose/compose-file/
- *Documentation: 15: PostgreSQL 15.1 Documentation*. (n.d.). PostgreSQL. Retrieved February 6, 2023, from https://www.postgresql.org/docs/current/