Práctica 3 - Docker

**Prof.:** Emma di Battista

**Estudiante:** Rommel Contreras

**Carnet:** 14-10242

*Para esta práctica trabajará con****el mismo repositorio git****que creó para la práctica anterior.*  
  
A su jefe en DeLab le gustó el inventario automatizado que extrae su script, y quiere que el CSV que genera su script este **siempre actualizado**, por lo que le ha pedido que ese script se ejecute en la plataforma docker en producción de la compañía.

1. Modifique el código de inventario de la asignación anterior para que la consulta se ejecute cada 5 minutos, y cada vez que se ejecute, actualice el archivo CSV generado. (Utilice el módulo "time" de python para el temporizador)

Para realizar este inciso se importó la librería *time* de *Python* y se agregó a la sección de imports de nuestro código anterior:

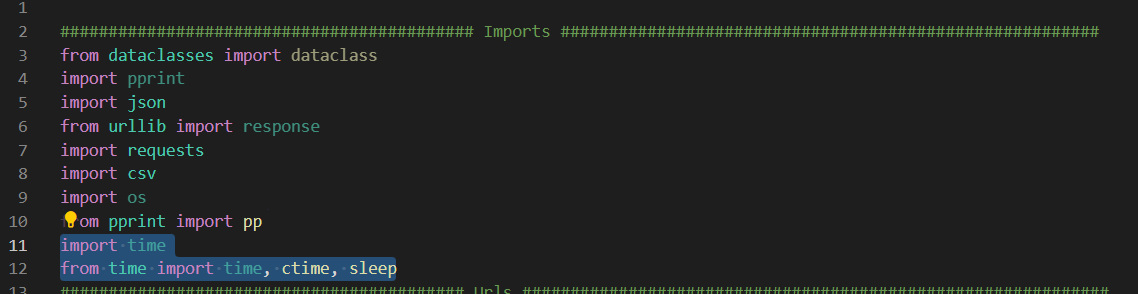


Ilustración 1 sección de imports de las librerias necesarias para el código DeLab.py

Luego se creó una función *countdown()* para realizar el contador de los 5 minutos la cual se muestra a continuación:

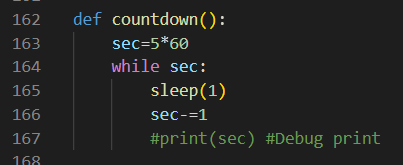


Ilustración 2 función countdown() para contar 5 minutos

Se cambió la aplicación principal que solicitaba al usuario una entrada y se adecuó a las necesidades del proyecto para que se hicieran las consultas a la API Cada 5 minutos.

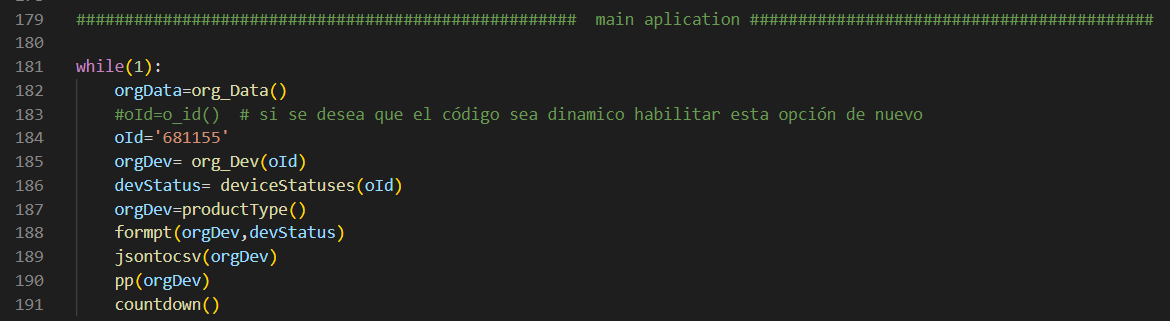


Ilustración 3 Nueva aplicación principal

2. Escriba un Dockerfile para crear un contenedor que ejecute el código python del punto anterior.

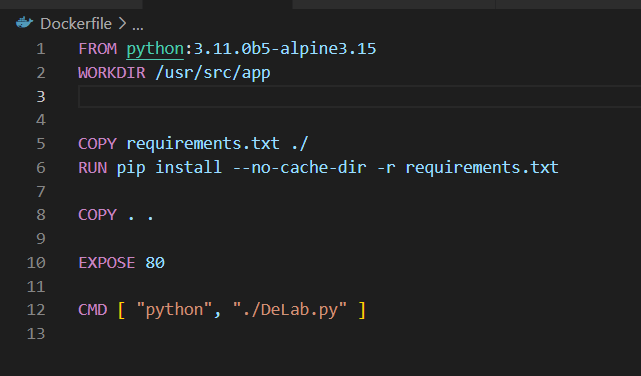


Ilustración 4 Dockerfile

3. Construya y ejecute el contenedor desde la línea de comandos.

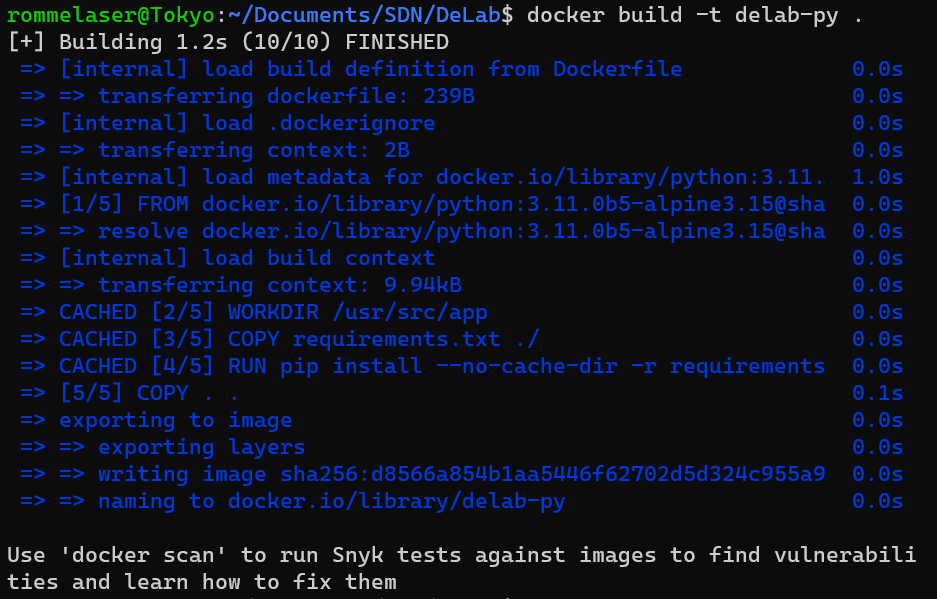


Ilustración 5 Construcción de la imagen delab-py

5. Haga un nuevo commit de su código añadiendo el Dockerfile en el repositorio que creó en la asignación anterior.

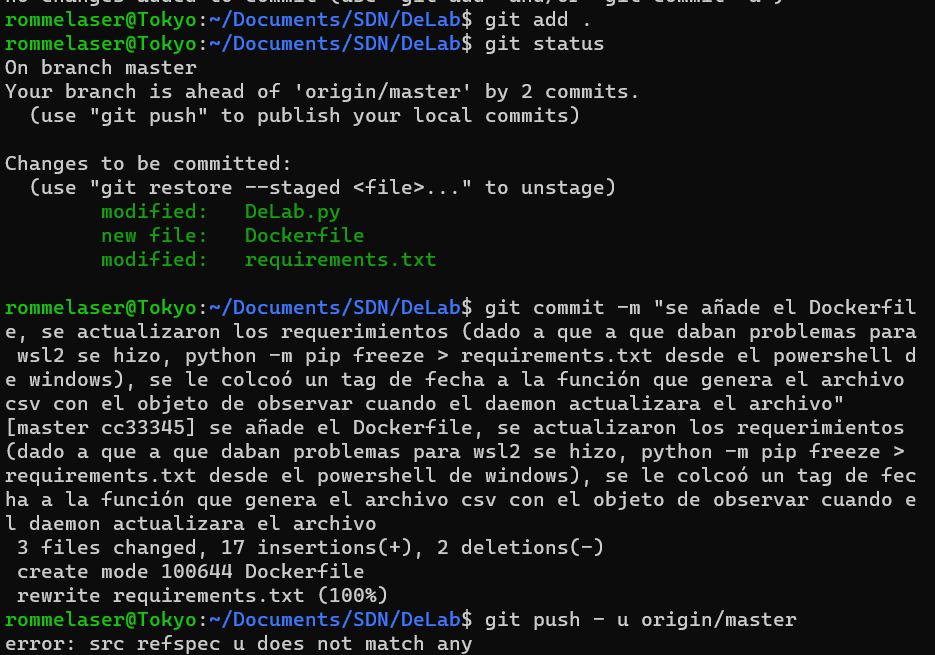


Ilustración 6 Commit inciso 5

6. Cree un volumen para su contenedor, en la dirección donde su código almacena el archivo CSV para hacer que este persista en memoria aún cuando se detenga el contenedor. (Modifique el Dockerfile de ser necesario)

Se puede hacer asi:

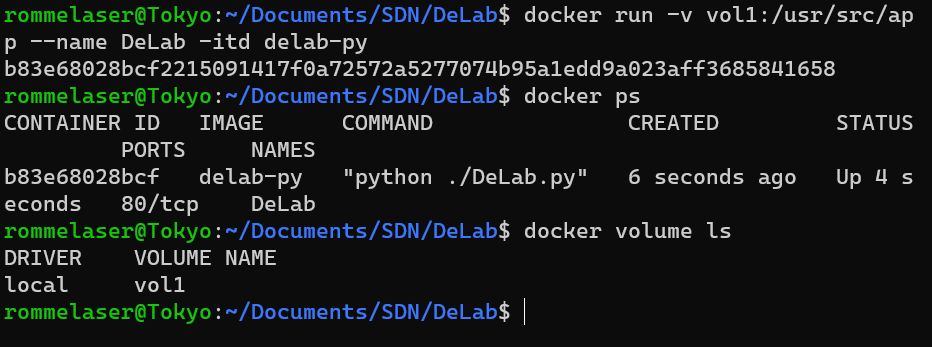


Ilustración 7 Creación del volumen y del contenedor incluyendo ejecución de la imagen como demonio con las opciones de interactividad terminal y trabajo en segundo plano (demonio)

O se puede modificar el Dockerfile

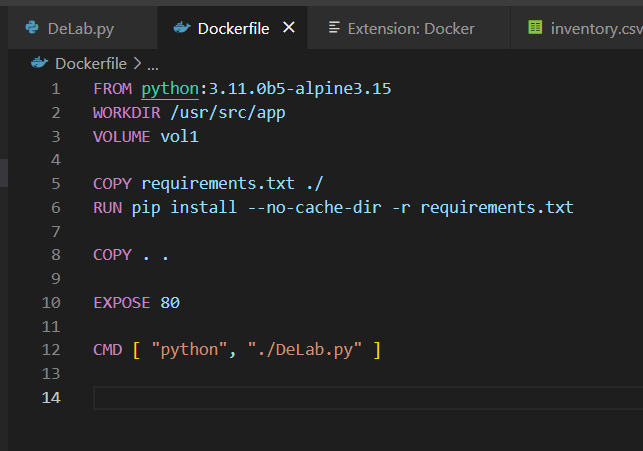


Ilustración 8 Creación del volumen vol1 desde el Dockerfile

La desventaja de esta es que el volumen queda con un nombre aleatorio

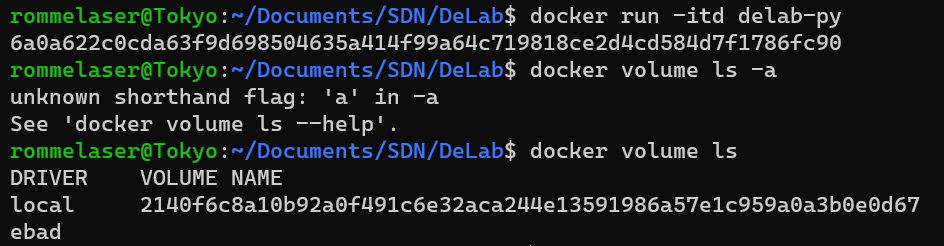


Ilustración 9 Ejemplo de que el volumen se crea con un nombre aleatorio

Por lo que es una mala práctica, así que se usará el otro método y si borrará este volumen.

7. Construya un archivo docker-compose.yml a partir del cual se pueda iniciar su contenedor con todos los volúmenes necesarios.

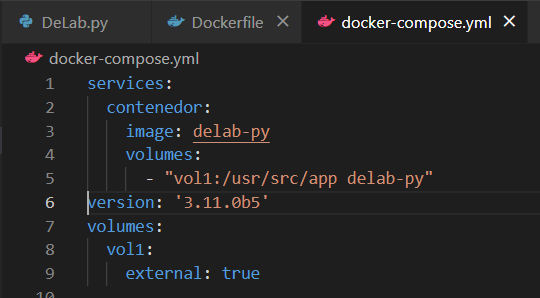


Ilustración 10 Archivo docker-compose.yml

8. Construya y ejecute su contenedor desde la línea de comandos usando el archivo docker compose que creó.

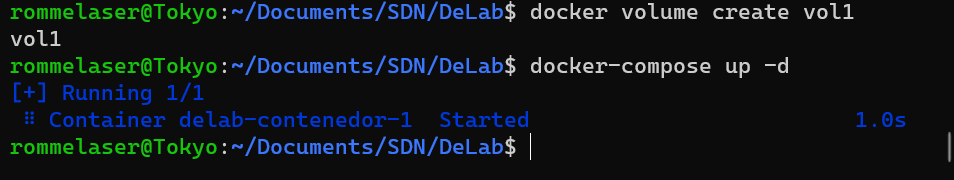
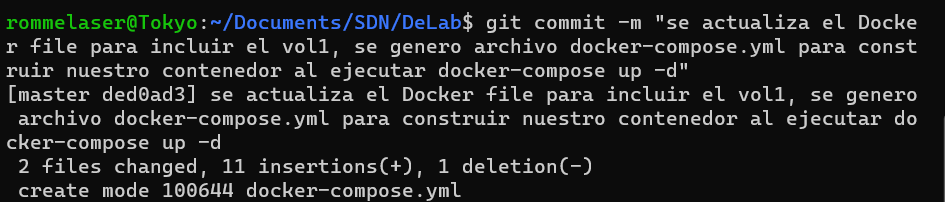


Ilustración 11 Iniciado del contenedor mediante docker-compose

9. Haga commit nuevamente a su repositorio, añadiendo el archivo docker-compose.

  
Ilustración 12 Commit para agregar el docker-compose.yml

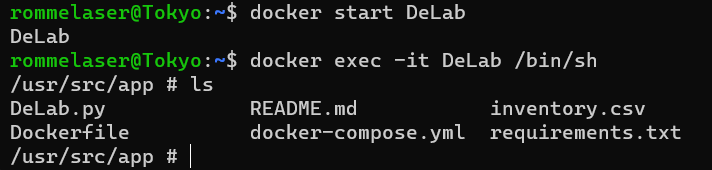
Se anexa una imagen con el contendor funcionando y conteniendo los archivos que deberíamos tener 

Ilustración 13 Contenedor en operación y con los archivos requeridos.