



ACTIVIDAD 08

GUIA DOCKER SPEED DOWNLOADER

MATERIA:

Computación tolerante a fallas (I7036)

NRC:

179961

SECCIÓN:

D06

MAESTRO:

López Franco, Michel Emmanuel

HORARIO:

Lunes y Miércoles, 11:00 – 12:55

FECHA DE ENTREGA:

19/04/2023

ALUMNO:

J. Emmanuel Ortiz Renteria (219747611)

Contenido

Introducción	3
Explicación del programa	3
Requerimientos	4
Preparación	4
Código de main.py:	5
Código de Dockerfile:.....	6

Introducción

Docker es una plataforma de código abierto diseñada para simplificar el desarrollo, el despliegue y la ejecución de aplicaciones mediante contenedores. Los contenedores son entornos ligeros y portátiles que encapsulan todo lo necesario para ejecutar una aplicación, incluidas las bibliotecas, las dependencias y el código, de manera que pueda ejecutarse de manera uniforme en cualquier entorno que admita Docker.

Usos de Docker:

1. **Desarrollo Ágil:** Docker facilita la creación de entornos de desarrollo reproducibles y consistentes, lo que acelera el ciclo de desarrollo y permite a los equipos trabajar de manera más eficiente y colaborativa.
2. **Despliegue de Aplicaciones:** Docker simplifica el proceso de despliegue de aplicaciones al empaquetarlas junto con todas sus dependencias en contenedores. Esto garantiza que las aplicaciones se ejecuten de manera consistente en cualquier entorno, desde el desarrollo hasta la producción.
3. **Microservicios:** Docker es ideal para la arquitectura de microservicios, donde las aplicaciones se descomponen en componentes más pequeños y modulares. Los contenedores de Docker permiten a los equipos desarrollar, desplegar y escalar fácilmente cada microservicio de forma independiente.
4. **Entornos de Pruebas y QA:** Docker facilita la creación de entornos de pruebas aislados y reproducibles, lo que permite a los equipos probar sus aplicaciones de manera eficiente y garantizar la calidad del software antes de su implementación en producción.
5. **Escalado Horizontal:** Docker facilita el escalado horizontal de aplicaciones al permitir la ejecución de múltiples instancias de contenedores de manera rápida y eficiente. Esto garantiza que las aplicaciones puedan manejar cargas de trabajo variables y escalen según sea necesario.
6. **Portabilidad y Consistencia:** Los contenedores de Docker ofrecen portabilidad y consistencia, lo que significa que una vez creados, los contenedores pueden ejecutarse de manera consistente en cualquier entorno que admita Docker, ya sea en un entorno local, en la nube o en un centro de datos. Esto simplifica la gestión de la infraestructura y garantiza que las aplicaciones se comporten de la misma manera en todos los entornos, lo que reduce los errores y mejora la eficiencia operativa.

Explicación del programa

Usando Docker se crea una imagen que instala temporalmente el modulo **request** en el contenedor para luego correr "main.py". El programa es simple, crea un thread que contiene la función para ejecutar un bucle infinito, dentro de este se llama otra función que realiza una descarga de datos a una dirección URL, y analiza el tiempo requerido para la descarga (final - inicio), y el peso de lo descargado. Mientras que en la consola

principal se entra otro bucle. El propósito de este es esperar una señal de paro por parte del teclado para activar el evento que detendrá el thread, mas sin antes acabar el ultimo trabajo, para cerrar sin problemas de manejo de datos y memoria.

Requerimientos

- Python 3+ Verifica que Python esté disponible en tu PATH para que puedas ejecutarlo desde la línea de comandos.
- Docker Desktop Docker está disponible para sistemas operativos Linux, macOS y Windows.
- Visual Studio Code.
- Dockerfile: El proyecto debe contener un Dockerfile que incluya todas las dependencias necesarias, incluyendo Python y el módulo **requests**.
- Sistema Operativo:
 - Windows 10 o superior
 - Linux
 - Mac Os

Preparación

1. Abrir Docker Desktop
2. Abrir Visual Studio Code
3. Instalar la extensión Docker para Visual Studio.
4. En Visual Studio crear una carpeta. Guardar main.py y Dockerfile en la carpeta.
5. En Visual Studio abrir nuevo terminal.
6. Escribir en la terminal el comando, app1 solo es el nombre que se le asigna al contenedor cambiarlo al gusto: `docker build -t app1 .`
Ver figura 1.
7. Escribir el comando: `docker run --rm -it app1`
8. Para cerrar el programa:
 - a. Hacer la combinación de teclas: Ctrl + C
 - b. Realizar el parado de contenedor desde Docker.

Código de main.py:

```
import threading
import time
import requests
import sys

class Descarga(threading.Thread):
    def __init__(self, url):
        threading.Thread.__init__(self)
        self.url = url

    def run(self):
        inicio = time.time()
        respuesta = requests.get(self.url)
        fin = time.time()
        tiempo_descarga = fin - inicio
        print(f"Descarga de {len(respuesta.content)} bytes completada en {tiempo_descarga:.2f} segundos")

def descarga(url, condicion):
    while not condicion.is_set():
        try:

print("\n=====")
        print("Revisando enlaces...")
        time.sleep(30) # Espera exactamente 30 segundos desde esta
seccion para volver a descargar los datos
        print("Realizando descarga...")
        descarga = Descarga(url)
        descarga.start()
        descarga.join()
        print("Descarga completada. Resultados impresos.")

print("=====")
        except Exception as e:
            condicion.set()
            print("\nError durante proceso. Cerrando programa. Mensaje de
error:\n")
            print(e)

print("=====")

if __name__ == "__main__":
    # URL de prueba para descarga
    enlace =
"https://github.com/JEmmanuelOR350/ID_random_generator/blob/c83176d026fc5
39e1ebf0027ce441fb33f36e19f/nombres.txt"

    condicion = threading.Event() #Evento para controlar la condición del
bucle
    descarga_thread = threading.Thread(target=descarga, args=(enlace,
condicion))
    descarga_thread.start()

    #Bucle para capturar una interrupccion por teclado, y arrojar el
cierre del thread
    try:
        while True:
            time.sleep(1)
```

```
except KeyboardInterrupt:
    condicion.set()
    print("\nDetención asistida por el usuario. Cerrando el
programa.")
    descarga_thread.join()
```

Código de Dockerfile:

```
FROM python
WORKDIR /app
COPY main.py .
RUN pip install --no-cache-dir requests
CMD ["python","main.py"]
```

Capturas de ejecución:

Figura 1. Visual Studio, resultados en la terminal después de ejecutar el comando `docker build -t app1 .`

Figura 2. Docker, visualización del contenedor creado para la imagen app1.

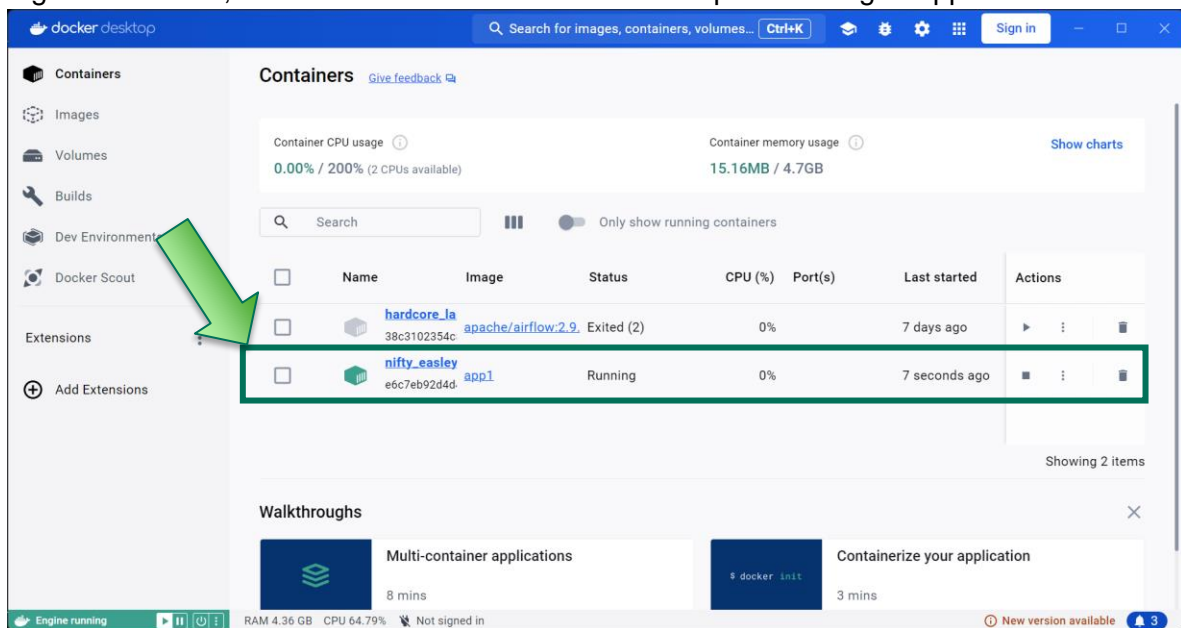


Figura 3. Docker, visualización de entradas y salidas de operaciones durante la corrida de la imagen.

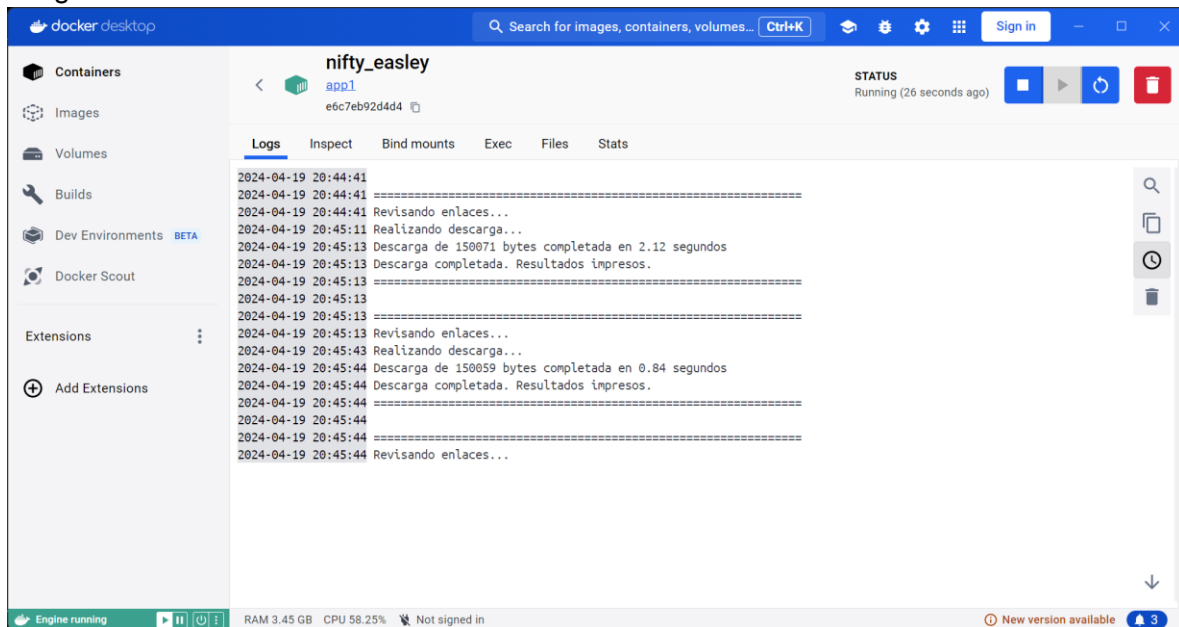


Figura 4. Contenedor Docker Desktop, Resultado de correr la aplicación hasta provocar la condición de paro.

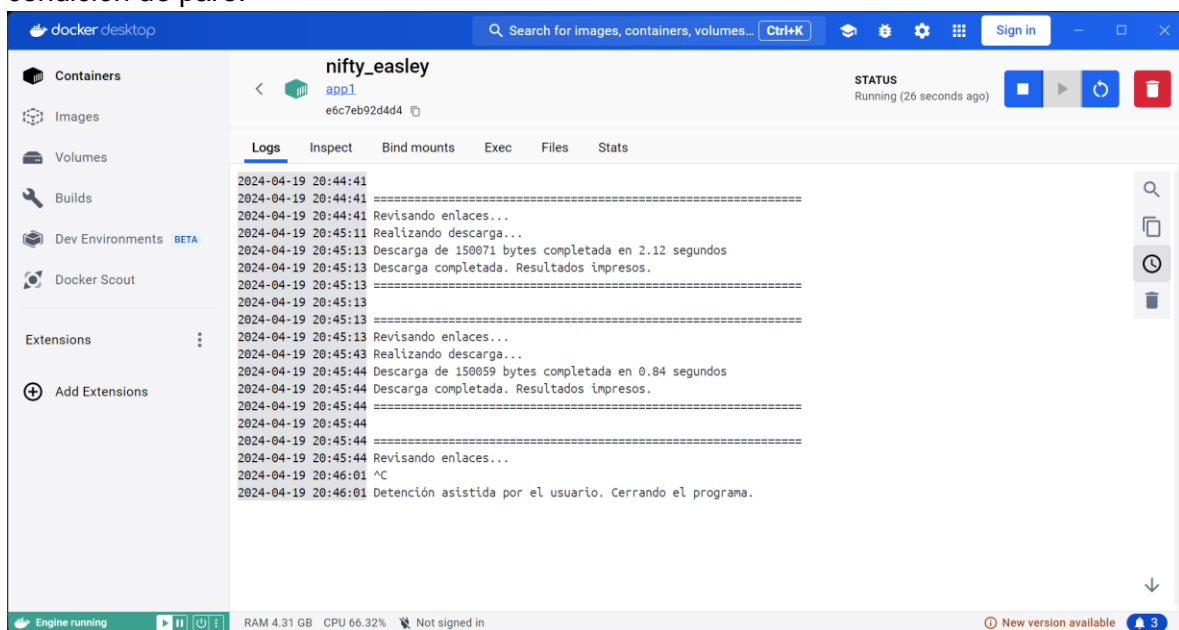
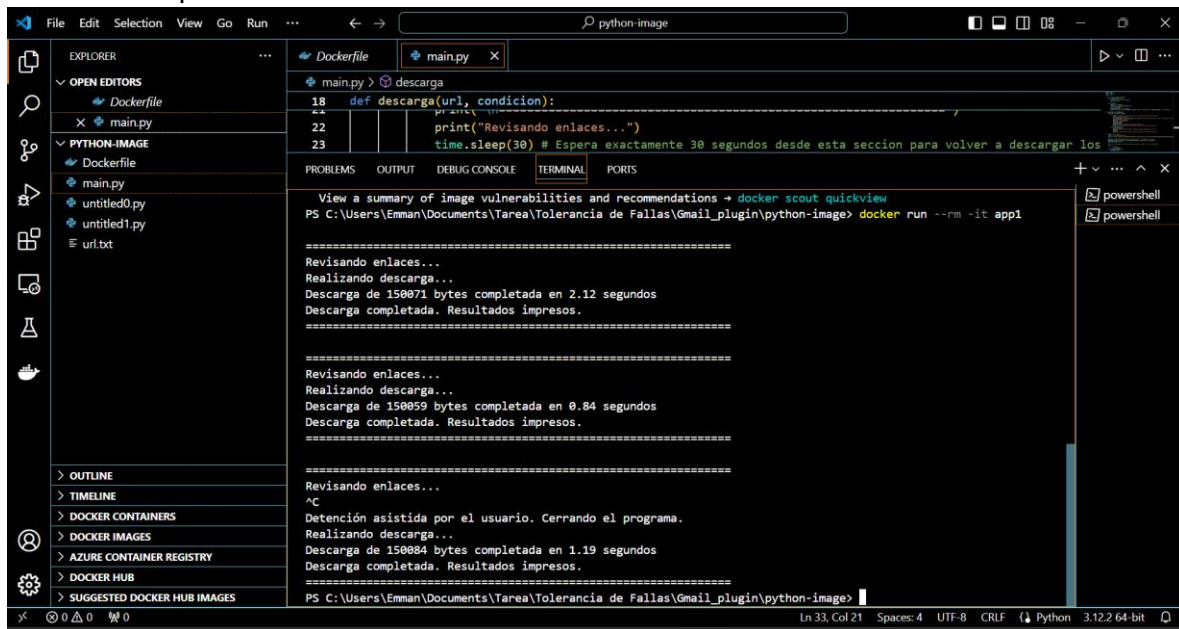


Figura 5. Terminal Visual Studio, Resultado de correr la aplicación hasta provocar la condición de paro.



```
File Edit Selection View Go Run ... python-image
EXPLORER
  OPEN EDITORS
    Dockerfile
    main.py
  PYTHON-IMAGE
    Dockerfile
    main.py
    untitled0.py
    untitled1.py
    url.txt
  OUTLINE
  TIMELINE
  DOCKER CONTAINERS
  DOCKER IMAGES
  AZURE CONTAINER REGISTRY
  DOCKER HUB
  SUGGESTED DOCKER HUB IMAGES

Dockerfile
main.py x
main.py > descarga
18 def descarga(url, condicion):
22     print("Revisando enlaces...")
23     time.sleep(30) # Espera exactamente 30 segundos desde esta seccion para volver a descargar los

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
View a summary of image vulnerabilities and recommendations + docker scout quickview
PS C:\Users\Emman\Documents\TareaTolerancia de Fallas\Gmail_plugin\python-image> docker run --rm -it appl

=====
Revisando enlaces...
Realizando descarga...
Descarga de 150071 bytes completada en 2.12 segundos
Descarga completada. Resultados impresos.
=====

=====
Revisando enlaces...
Realizando descarga...
Descarga de 150059 bytes completada en 0.84 segundos
Descarga completada. Resultados impresos.
=====

=====
Revisando enlaces...
^C
Detención asistida por el usuario. Cerrando el programa.
Realizando descarga...
Descarga de 150084 bytes completada en 1.19 segundos
Descarga completada. Resultados impresos.
=====

PS C:\Users\Emman\Documents\TareaTolerancia de Fallas\Gmail_plugin\python-image>
```