Centro Universitario de Ciencias Exactas e Ingenierías

CUCEI

Computación Tolerante a Fallas

Diferencias entre Docker, Kubernetes, Apache Mesos y OpenShift

Michel Emanuel López Franco

Gutiérrez Galán Ruben Alejandro

Código: 214798315

Contenido

¿Qué es Docker?:

Docker es una plataforma de software que permite a los desarrolladores empaquetar, distribuir y ejecutar aplicaciones de manera consistente en diferentes entornos. Utiliza la tecnología de contenedores para encapsular una aplicación y sus dependencias en una unidad llamada "contenedor". Estos contenedores son portátiles y pueden ejecutarse en cualquier máquina que tenga Docker instalado, independientemente de las diferencias en el entorno de desarrollo, la infraestructura o la configuración del sistema operativo.

Algunas características clave de Docker incluyen:

- Contenedores: Los contenedores son instancias ligeras y eficientes que contienen una aplicación y todas sus dependencias, incluidas bibliotecas y configuraciones. Se pueden ejecutar en cualquier entorno que admita Docker, lo que facilita la implementación y la escalabilidad.
- Imagen: Una imagen de Docker es un paquete independiente que incluye todo lo necesario para ejecutar una aplicación, incluidos el código, las bibliotecas y las configuraciones. Las imágenes se utilizan como base para crear contenedores.
- Dockerfile: Un Dockerfile es un archivo de configuración que especifica cómo construir una imagen de Docker. Define las instrucciones para instalar dependencias, configurar el entorno y copiar el código fuente.
- Orquestación: Docker proporciona herramientas de orquestación, como Docker Compose y Kubernetes, que permiten gestionar y coordinar la implementación de aplicaciones en múltiples contenedores.
- Portabilidad: Debido a la naturaleza de los contenedores, las aplicaciones empaquetadas con Docker son altamente portátiles y pueden ejecutarse de manera consistente en diferentes entornos, desde el desarrollo hasta la producción.

¿Qué es Kubernetes?:

Kubernetes es una plataforma de código abierto diseñada para automatizar la implementación, escalado y gestión de aplicaciones contenerizadas. Fue desarrollado por Google y luego donado a la Cloud Native Computing Foundation (CNCF). Kubernetes facilita la orquestación de contenedores, lo que significa que coordina la ejecución de contenedores en un clúster de máquinas, asegurándose de que las aplicaciones se ejecuten de manera eficiente y confiable en entornos de producción.

Algunas características clave de Kubernetes incluyen:

- Orquestación de Contenedores: Kubernetes simplifica y automatiza tareas como la implementación, escalado y administración de contenedores. Permite a los desarrolladores definir cómo se deben ejecutar y escalar sus aplicaciones, así como manejar la resiliencia y la autorreparación.
- Escalabilidad: Kubernetes facilita la escalabilidad de las aplicaciones al permitir la adición o eliminación dinámica de instancias de contenedores según la demanda. Esto garantiza que la aplicación pueda manejar diferentes cargas de trabajo sin intervención manual.
- Gestión de Recursos: Controla y asigna recursos de hardware a los contenedores, lo que ayuda a optimizar el rendimiento y la eficiencia en el uso de recursos.

- Automatización de Despliegues: Kubernetes permite la implementación continua y la entrega continua (CI/CD) mediante la automatización de procesos, lo que facilita la actualización de aplicaciones y la implementación de nuevas versiones sin tiempo de inactividad.
- Monitoreo y Registro: Proporciona herramientas para monitorear la salud de las aplicaciones y recopilar registros, lo que facilita la detección y resolución de problemas.
- Service Discovery y Balanceo de Carga: Facilita la exposición de servicios y la gestión del tráfico entre ellos, asegurando la disponibilidad y la distribución equitativa de las solicitudes.
- Declarativo: Kubernetes utiliza una configuración declarativa a través de archivos YAML, lo que significa que los usuarios describen el estado deseado de sus aplicaciones y Kubernetes se encarga de llevar el sistema al estado deseado.

¿Qué es Apache Mesos?:

Apache Mesos es un sistema de código abierto diseñado para gestionar y escalar aplicaciones distribuidas en grandes clústeres de máquinas. Fue desarrollado originalmente en la Universidad de California, Berkeley, y posteriormente se convirtió en un proyecto de la Apache Software Foundation. Mesos proporciona una abstracción de recursos a través de múltiples nodos en un clúster, permitiendo que las aplicaciones aprovechen eficientemente los recursos disponibles.

Algunas características clave de Apache Mesos son:

- Abstracción de Recursos: Mesos ofrece una capa de abstracción que permite a las aplicaciones tratar todo el clúster como un único recurso colectivo. Puede ejecutar múltiples marcos de trabajo (frameworks) en un clúster, y cada uno de ellos puede solicitar y utilizar recursos de manera independiente.
- Escalabilidad: Mesos está diseñado para escalar horizontalmente, lo que significa que puede manejar grandes clústeres de máquinas y adaptarse a cambios en la carga de trabajo.
- Tolerancia a Fallos: Mesos es resistente a fallos y puede recuperarse automáticamente de situaciones como la pérdida de nodos en el clúster.
- Soporte para Múltiples Frameworks: Admite varios frameworks de aplicaciones, incluyendo Apache Hadoop, Apache Spark y Marathon. Cada framework puede tener sus propios requisitos y políticas de programación.
- Arquitectura Maestro-Esclavo: La arquitectura de Mesos se basa en un componente maestro (master) que coordina la asignación de recursos y en varios nodos esclavos (slave) que ejecutan las tareas de las aplicaciones.
- Programación Dinámica de Recursos: Mesos permite la asignación dinámica de recursos en función de la carga de trabajo, lo que mejora la eficiencia en la utilización de recursos.
- Soporte para Contenedores: Mesos tiene soporte nativo para contenedores, lo que significa que puede ejecutar aplicaciones empaquetadas en contenedores, como Docker.

¿Qué es OpenShift?:

OpenShift es una plataforma de contenedores de código abierto desarrollada por Red Hat. Proporciona herramientas para desarrollar, implementar y administrar aplicaciones contenerizadas. OpenShift se basa en la tecnología de contenedores Docker y la orquestación de contenedores

Kubernetes, y agrega funcionalidades y herramientas adicionales para simplificar el ciclo de vida completo de desarrollo y operaciones.

Algunas características clave de OpenShift incluyen:

- Orquestación de Contenedores: OpenShift utiliza Kubernetes como su motor de orquestación de contenedores, lo que facilita la implementación, escalado y gestión de aplicaciones en entornos contenerizados.
- Desarrollo de Aplicaciones: Proporciona herramientas integradas para el desarrollo de aplicaciones, incluyendo integración con entornos de desarrollo integrado (IDE), pipelines de CI/CD (implementación continua y entrega continua), y soporte para varias tecnologías de desarrollo.
- Plataforma de Aplicaciones Empaquetadas: OpenShift facilita la implementación de aplicaciones en forma de contenedores, lo que permite una mayor portabilidad y consistencia en diferentes entornos.
- Escalabilidad y Tolerancia a Fallos: Al heredar las capacidades de Kubernetes, OpenShift permite la escalabilidad horizontal y proporciona tolerancia a fallos para las aplicaciones.
- Seguridad Integrada: Incluye características de seguridad integradas, como autenticación y autorización, escaneo de vulnerabilidades de contenedores, y políticas de seguridad para proteger las aplicaciones y los datos.
- Monitoreo y Registro: Ofrece herramientas integradas para el monitoreo del rendimiento de las aplicaciones y la recopilación de registros, lo que facilita la resolución de problemas y la optimización del rendimiento.
- Administración de Recursos: Permite una gestión eficiente de recursos, incluida la asignación dinámica y el control de recursos para aplicaciones.
- Despliegue en Entornos Multinube: OpenShift puede implementarse en entornos on-premise, en la nube pública o en entornos híbridos, brindando flexibilidad a las organizaciones en cuanto a dónde ejecutan sus aplicaciones.

¿Qué es Rancher?:

Rancher es una plataforma de gestión de contenedores de código abierto que facilita la implementación, administración y escalado de contenedores en entornos empresariales. Rancher proporciona una interfaz unificada y herramientas para orquestar y gestionar contenedores en clústeres de Kubernetes, así como en otros entornos de contenedores como Docker Swarm y Apache Mesos.

Algunas características clave de Rancher incluyen:

- Administración de Clústeres: Rancher permite crear, administrar y escalar clústeres de contenedores en entornos locales, en la nube o en entornos híbridos. Admite múltiples proveedores de nube, como AWS, Azure, Google Cloud, entre otros.
- Orquestación de Kubernetes: Ofrece soporte integral para Kubernetes, permitiendo la implementación y gestión sencilla de clústeres Kubernetes. También puede gestionar clústeres de Docker Swarm y Mesos.

- Interfaz de Usuario Gráfica (GUI): Proporciona una interfaz gráfica fácil de usar que simplifica tareas como la implementación de aplicaciones, la supervisión del estado del clúster y la administración de recursos.
- Implementación y Gestión de Aplicaciones: Facilita la implementación de aplicaciones mediante la definición de stacks (conjuntos de servicios y configuraciones) a través de archivos de configuración y plantillas.
- Escalabilidad y Alta Disponibilidad: Rancher facilita la escalabilidad de las aplicaciones y los servicios, permitiendo la adición y el escalado de nodos de manera dinámica. También proporciona opciones para garantizar la alta disponibilidad de las aplicaciones.
- Seguridad: Ofrece características de seguridad, como la gestión de roles y permisos, la integración con proveedores de identidad externos, y el cifrado de datos en tránsito.
- Registro y Monitoreo: Proporciona herramientas para la recopilación de registros y el monitoreo del rendimiento de los clústeres y las aplicaciones.
- Extensibilidad: Rancher es extensible y permite la integración con otras herramientas y sistemas, lo que facilita la personalización según las necesidades específicas de la organización.

¿Qué es Istio?:

Istio es una plataforma de servicios de código abierto que se utiliza para conectar, gestionar y asegurar microservicios en aplicaciones contenerizadas. Proporciona una capa de servicios de red que permite la comunicación segura y eficiente entre servicios distribuidos, así como el control y la observabilidad de tráfico entre estos servicios.

Algunas características clave de Istio incluyen:

- Gestión de Tráfico: Istio facilita la gestión del tráfico entre microservicios, permitiendo la aplicación de reglas de enrutamiento y políticas de tráfico. Esto incluye la capacidad de realizar divisiones de tráfico para pruebas A/B, canarios y otras estrategias de implementación.
- Seguridad de Servicios: Proporciona funciones de seguridad para microservicios, como la autenticación y autorización entre servicios, el cifrado de datos en tránsito y la gestión de certificados.
- Monitoreo y Observabilidad: Istio recopila datos de telemetría y proporciona herramientas para monitorear y observar el comportamiento de los servicios en tiempo real. Esto incluye métricas, registros y trazas para facilitar la detección y resolución de problemas.
- Gestión de Políticas: Permite la aplicación de políticas de control de acceso, cuotas de tráfico y límites de velocidad, lo que mejora la seguridad y la gestión de recursos en un entorno de microservicios.
- Resiliencia: Istio incluye características para mejorar la resiliencia de las aplicaciones, como la tolerancia a fallos, la reintentación de solicitudes y la gestión de circuitos.
- Gestión de Identidad y Autorización: Proporciona servicios de gestión de identidad y autorización, permitiendo la autenticación de servicios y la toma de decisiones de autorización basada en políticas.

- Plataforma Independiente: Istio es una plataforma independiente que puede utilizarse con orquestadores de contenedores como Kubernetes y también en entornos de máquinas virtuales.
- Extensibilidad: Istio es extensible y admite la integración con otras herramientas y sistemas, así como la posibilidad de personalizar las políticas y configuraciones según las necesidades específicas.

Diferencias generales entre las diferentes herramientas:

1. Docker:
   - Docker es principalmente una plataforma para empaquetar, distribuir y ejecutar aplicaciones en contenedores.
   - Proporciona una tecnología de contenedorización que permite a las aplicaciones y sus dependencias ejecutarse de manera aislada y portátil.
   - Docker se utiliza comúnmente para el desarrollo y la implementación de aplicaciones en entornos de contenedores.
2. Kubernetes:
   - Kubernetes es una plataforma de orquestación de contenedores que automatiza la implementación, escalado y gestión de aplicaciones contenerizadas.
   - Se centra en la orquestación y gestión de clústeres de contenedores, permitiendo la ejecución eficiente de aplicaciones distribuidas.
   - Proporciona una abstracción de recursos, lo que facilita la administración de aplicaciones en entornos de producción.
3. Apache Mesos:
   - Mesos es un sistema de gestión de clústeres que proporciona una abstracción de recursos a través de múltiples nodos en un clúster.
   - Se utiliza para gestionar y escalar aplicaciones distribuidas, y es compatible con varios frameworks de aplicaciones, incluidos Apache Hadoop y Apache Spark.
   - Mesos no se centra exclusivamente en contenedores, aunque tiene soporte para la ejecución de tareas en contenedores.
   - Proporciona una arquitectura maestro-esclavo, donde el maestro coordina la asignación de recursos y los nodos esclavos ejecutan tareas.
   - Mesos se centra en la eficiencia y la utilización de recursos, permitiendo la ejecución de diversas cargas de trabajo en el mismo clúster.
   - Aunque no es tan común en la gestión de contenedores como Kubernetes, puede integrarse con orquestadores de contenedores para trabajar en conjunto.
4. OpenShift:
   - OpenShift es una plataforma de contenedores de Red Hat que se basa en Kubernetes y agrega herramientas y funcionalidades adicionales.
   - Ofrece un conjunto integrado de herramientas para el desarrollo, implementación y administración de aplicaciones contenerizadas.
   - Incluye características adicionales como gestión de roles y permisos, seguridad mejorada y capacidades de desarrollo integrado.
5. Rancher:

- Rancher es una plataforma de gestión de contenedores que soporta múltiples orquestadores, incluyendo Kubernetes, Docker Swarm y Apache Mesos.
- Facilita la administración unificada de clústeres de contenedores y proporciona una interfaz gráfica para simplificar tareas operativas.
- Es conocido por su flexibilidad y capacidad para gestionar entornos contenerizados heterogéneos.

6. Istio:
   - Istio es una plataforma de servicios que se utiliza para conectar, gestionar y asegurar microservicios en aplicaciones contenerizadas.
   - Se centra en mejorar la comunicación entre servicios distribuidos, proporcionando características como gestión de tráfico, seguridad, y monitoreo.
   - Suele integrarse con orquestadores de contenedores como Kubernetes para mejorar las capacidades de servicio de las aplicaciones.

Ejemplo de programa desarrollado:

Se hará uso del codigo desarrollado para la actividad de kubernetes, en lugar de usar Docker desktop usaremos Rancher desktop para esta actividad y verificaremos si hay cambios muy significativos.

El codigo desarrollado es el siguiente:

app.py: codigo desarrollado que crea una calculadora básica, está desarrollado en Python y usa flask para su funcionamiento.

```
# app.py
from flask import Flask, request, jsonify

app = Flask(__name__)

def guardar_resultado(resultado_decimal, resultado_hexa):
    with open('resultados.txt', 'a') as file:
        file.write(f"Decimal: {resultado_decimal}, Hexadecimal:
{resultado_hexa}\n")
    file.close()

@app.route('/calcular', methods=['POST'])
def calcular():
    data = request.get_json()

    operacion = data['operacion']
    numero1 = data['numero1']
    numero2 = data['numero2']

    if operacion == 'sumar':
        resultado = numero1 + numero2
    elif operacion == 'restar':
        resultado = numero1 - numero2
```

```
    elif operacion == 'multiplicar':
        resultado = numero1 * numero2
    elif operacion == 'dividir':
        if not isinstance(numero1, (int, float)) or not isinstance(numero2,
(int, float)):
            return jsonify({"error": "Los números deben ser enteros o de
punto flotante"}), 400
        if numero2 != 0:
            resultado = numero1 / numero2
        else:
            return jsonify({"error": "No se puede dividir por cero"}), 400
    else:
        return jsonify({"error": "Operación no válida"}), 400
    resultado=int(resultado)
    resultado_hexa = hex(resultado)  # Convierte el resultado a hexadecimal
    guardar_resultado(resultado, resultado_hexa)  # Guarda el resultado en
el archivo

    return jsonify({"resultado": resultado, "resultado_hexa":
resultado_hexa})


if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

Dockerfile: archive dockerfile que nos permitirá configurar la aplicación con las especificaciones necesarias para el funcionamiento del programa de manera correcta.

```
# Dockerfile
# utiliza una imagen base de python
FROM python:3.8-slim

# establece el directorio de trabajo
WORKDIR /app
# copia los archivos locales al contenedor
COPY app.py .
COPY requiriments.txt .

# instalar las dependencias
RUN pip install -r requiriments.txt

#exponer el puerto en el que se usa la aplicacion Flask
EXPOSE 5000
```

```
#ejecuta la aplicacion al iniciar el contenedor
CMD ["python","app.py"]
```

requiriments.txt: contiene las especificaciones para flask.

Flask==2.0.1

werkzeug==2.0.1

Comandos usados para el desarrollo:

docker build -t mi_aplicacion_python:

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  COMENTARIOS

    from .helpers import get_debug_flag
  File "/usr/local/lib/python3.8/site-packages/flask/helpers.py", line 16, in <module>
    from werkzeug.urls import url_quote
ImportError: cannot import name 'url_quote' from 'werkzeug.urls' (/usr/local/lib/python3.8/site-packages/werkzeug/urls.py)
PS D:\Programacion\Classroom\Computacion_Tolerante_a_fallas_1\Diferencias entre Docker, Kubernetes, Apache Mesos y OpenShift\Proyecto> d
ocker
build -t mi_aplicacion_python .
2023/11/27 14:31:50 http2: server: error reading preface from client //./pipe/docker_engine: file has already been closed
[+] Building 8.4s (10/10) FINISHED                                                                                    docker:de
fault
 => [internal] load .dockerignore
 0.1s
 => => transferring context: 2B
 0.0s
 => [internal] load build definition from Dockerfile
 0.0s
 => => transferring dockerfile: 467B
 0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim
 2.4s
 => [1/5] FROM docker.io/library/python:3.8-slim@sha256:19e07fa24813e88b04e606772213bd03ba044637cc939a211e28ccf997a9162a
 0.0s
 => => transferring context: 102B
 0.0s
 => CACHED [2/5] WORKDIR /app
 0.0s
 => CACHED [3/5] COPY app.py .
 0.0s
 => [5/5] RUN pip install -r requiriments.txt
 5.4s
 => exporting to image
 0.3s
 => => exporting layers
 0.3s
```

Docker images:

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  COMENTARIOS                                    powershell

PS D:\Programacion\Classroom\Computacion_Tolerante_a_fallas_1\Diferencias entre Docker, Kubernetes, Apache Mesos y OpenShift> cd Proyecto
PS D:\Programacion\Classroom\Computacion_Tolerante_a_fallas_1\Diferencias entre Docker, Kubernetes, Apache Mesos y OpenShift\Proyecto> docker images
REPOSITORY                                    TAG                 IMAGE ID       CREATED         SIZE
mi_aplicacion_python                          latest              9f81662589f5   2 hours ago     139MB
<none>                                        <none>              01c56bbf9460   2 hours ago     139MB
app.py                                        latest              d43de880de7c   2 hours ago     128MB
rancher/mirrored-library-traefik              2.10.5              cc365cbb0397   6 weeks ago     151MB
rancher/klipper-helm                          v0.8.2-build20230815  5f89cb8137cc   3 months ago    256MB
rancher/mirrored-library-busybox              1.36.1              a416a98b71e2   4 months ago    4.26MB
rancher/klipper-lb                            v0.4.4              af74bd845c4a   6 months ago    12MB
rancher/local-path-provisioner                v0.0.24             b29384aeb4b1   8 months ago    40.1MB
rancher/mirrored-metrics-server               v0.6.3              817bbe3f2e51   8 months ago    68.9MB
rancher/mirrored-coredns-coredns              1.10.1              ead0a4a53df8   9 months ago    53.6MB
rancher/mirrored-pause                        3.6                 6270bb605e12   2 years ago     683kB
ghcr.io/rancher-sandbox/rancher-desktop/rdx-proxy  latest              fccbed615a6d   N/A             5.12MB
PS D:\Programacion\Classroom\Computacion_Tolerante_a_fallas_1\Diferencias entre Docker, Kubernetes, Apache Mesos y OpenShift\Proyecto>
```

Docker run --rm -p 5000:5000 mi_aplicacion_python:

```
PS D:\Programacion\Classroom\Computacion_Tolerante_a_fallas_1\Diferencias entre Docker, Kubernetes, Apache Mesos y OpenShift\Proyecto> docker run --rm -p 5000:5000 mi_aplicacion_python
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.17.0.2:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 120-456-561
```

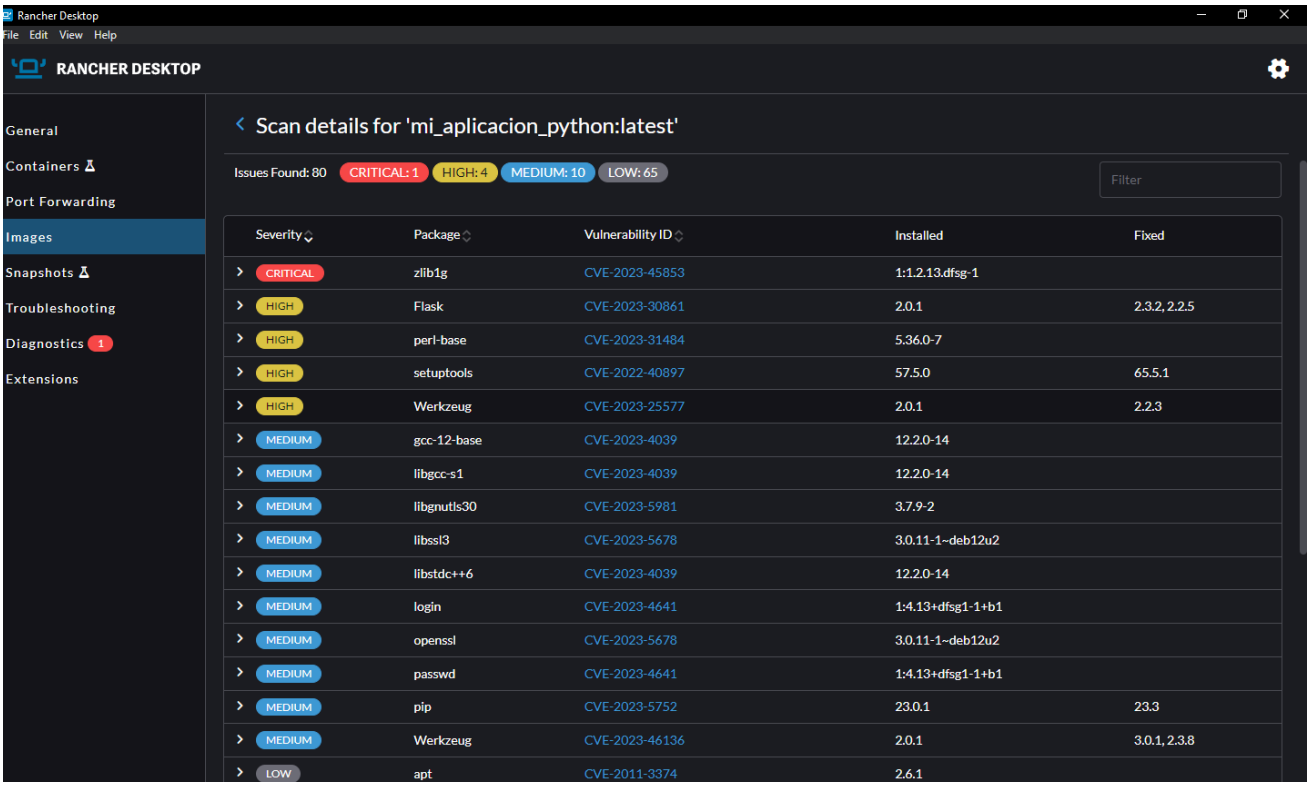
Ejecución del programa:

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  COMENTARIOS

PS D:\Programacion\Classroom\Computacion_Tolerante_a_fallas_1\Diferencias entre Docker, Kubernetes, Apache Mesos y OpenShift> Invoke-RestMethod -Uri "http://localhost:5000/calcular" -method Post -Headers @{"Content-Type"="application/json"} -Body '{"operacion":"multiplicar","numero1":1000,"numero2":5}'

resultado resultado_hexa
--------- --------------
     5000 0x1388


PS D:\Programacion\Classroom\Computacion_Tolerante_a_fallas_1\Diferencias entre Docker, Kubernetes, Apache Mesos y OpenShift>
```



Conclusión:

El desarrollo fue prácticamente igual a la actividad anterior y sus resultados fueron los mismos, fue bastante sencillo, la razón por la que use Rancher fue debido a que las otras herramientas no contaban con el soporte necesario para Windows o por lo menos eso era lo que me indicaban algunas páginas, sin embargo Rancher si que cuenta con soporte además de que este es usado de manera hibrida tanto para sistemas como Windows como con aquellos basados en Unix tal cual Linux o por ejemplo Mac-os que si cuentan con buen soporte, pero en Windows por lo menos con las versiones de mas casuales como lo es la 10 no tienen mucho soporte, en cambio Windows server parece tener mejor soporte para estas mismas.

Bibliografía:

- *Rancher Desktop by SUSE*. (n.d.). https://rancherdesktop.io/
- *Docker: Accelerated Container Application Development*. (2023e, October 18). Docker. https://www.docker.com/
- *Red Hat OpenShift enterprise Kubernetes container platform*. (n.d.). https://www.redhat.com/en/technologies/cloud-computing/openshift
- *Apache mesos*. (n.d.-b). Apache Mesos. https://mesos.apache.org/
- *Istio*. (n.d.-b). Istio. https://istio.io/