

📄 **Título del trabajo:** Virtualización

Alumnos:

Bruno Croppi - croppibruno@gmail.com

Ignacio Figueroa - ignaciofigueroadev@gmail.com

Materia: Arquitectura y Sistemas Operativos

📖 | **Índice**

1. Introducción
2. Marco Teórico
3. Caso Práctico
4. Metodología Utilizada
5. Resultados Obtenidos
6. Conclusiones
7. Bibliografía
8. Anexos

1. Introducción

En este trabajo práctico nos enfocamos en la **virtualización mediante contenedores**, utilizando **Docker** como tecnología principal. Esta herramienta nos permite ejecutar aplicaciones de forma aislada, liviana y portable, sin necesidad de una máquina virtual completa.

El objetivo fue desarrollar una calculadora de índice de masa corporal que se pueda contenerizar y ejecutar fácilmente en cualquier entorno con Docker instalado.

2. Marco Teórico

Docker es una plataforma de contenedorización que permite empaquetar aplicaciones junto con sus dependencias en contenedores. A diferencia de la virtualización tradicional, los contenedores no requieren un sistema operativo completo por cada instancia, lo cual mejora el rendimiento y la eficiencia.

Conceptos clave:

- **Contenedor:** entorno ligero y aislado para ejecutar software.
- **Imagen:** plantilla inmutable que contiene todo lo necesario para correr una app.
- **Dockerfile:** archivo que define cómo construir una imagen.
- **Docker CLI:** herramienta de línea de comandos para interactuar con Docker

3. Caso Práctico

Desarrollamos una **calculadora de IMC (Índice de Masa Corporal)** utilizando Python. Esta aplicación recibe el peso y la altura como argumentos y calcula el IMC, clasificando el resultado (bajo peso, normal, sobrepeso u obesidad):.

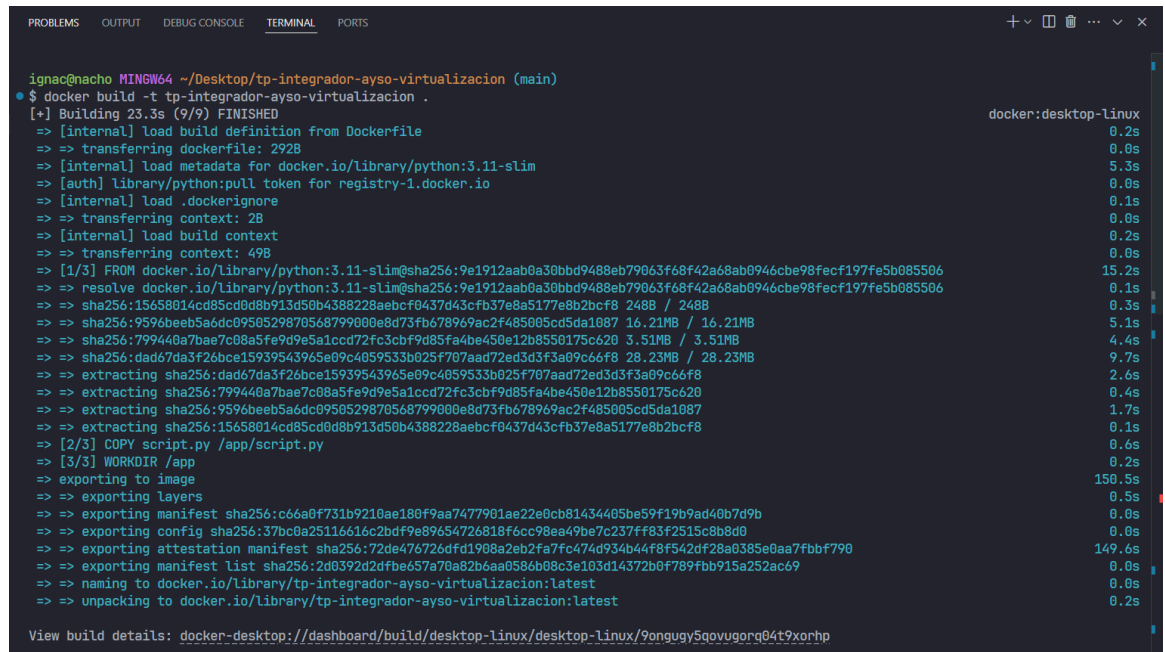
Luego, construimos una imagen de Docker usando un **Dockerfile** que contiene todo lo necesario para ejecutar el script.

Pasos clave:

1. Creamos el archivo [script.py](#)
2. Construimos la imagen de docker con

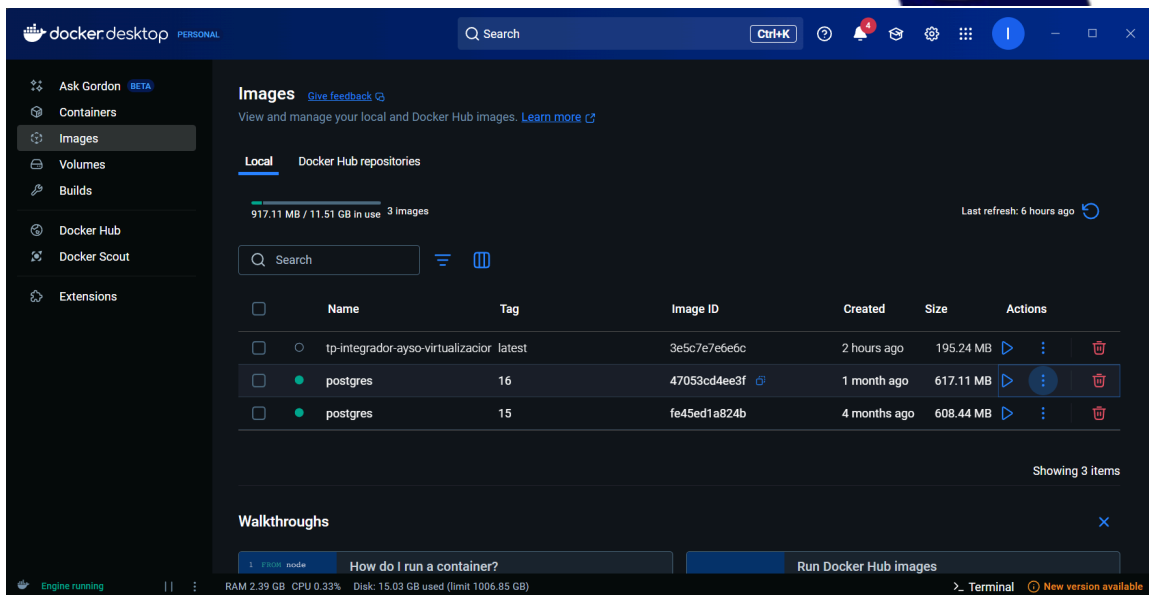
```
docker build -t tp-integrador-ayso-virtualizacion .
```

Imagen de la construcción de la imagen



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
ignacio@nacho MINGW64 ~/Desktop/tp-integrador-ayso-virtualizacion (main)
$ docker build -t tp-integrador-ayso-virtualizacion .
[+] Building 23.3s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 292B
=> [internal] load metadata for docker.io/library/python:3.11-slim
=> [auth] library/python:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 49B
=> [1/3] FROM docker.io/library/python:3.11-slim@sha256:9e1912aab0a30bbd9488eb79063f68f42a68ab0946cbe98fecf197fe5b085506
=> => resolve docker.io/library/python:3.11-slim@sha256:9e1912aab0a30bbd9488eb79063f68f42a68ab0946cbe98fecf197fe5b085506
=> => sha256:15658014cd85cd0d8b913d50b4388228aebcf0437d43cfb37e8a5177e8b2bcf8 248B / 248B
=> => sha256:9594beeb5a6dc0950529870568799000e8d73fb678969ac2f485005cd5da1087 16.21MB / 16.21MB
=> => sha256:799440a7bae7c08a5fe9d9e5a1ccd72fc3cbf9d85fa4be450e12b8550175c620 3.51MB / 3.51MB
=> => sha256:dad67da3f26bce15939543965e09c4059533b025f707aad72ad3d3f3a09c66f8 28.23MB / 28.23MB
=> => extracting sha256:dad67da3f26bce15939543965e09c4059533b025f707aad72ad3d3f3a09c66f8 2.6s
=> => extracting sha256:799440a7bae7c08a5fe9d9e5a1ccd72fc3cbf9d85fa4be450e12b8550175c620 0.4s
=> => extracting sha256:9594beeb5a6dc0950529870568799000e8d73fb678969ac2f485005cd5da1087 1.7s
=> => extracting sha256:15658014cd85cd0d8b913d50b4388228aebcf0437d43cfb37e8a5177e8b2bcf8 0.1s
=> [2/3] COPY script.py /app/script.py
=> => 0.6s
=> [3/3] WORKDIR /app
=> => 0.2s
=> => exporting layers 150.5s
=> => exporting manifest sha256:c66a0f731b9210ae180f9aa7477901ae22e0cb81434405be59f19b9ad40b7d9b 0.5s
=> => exporting config sha256:37bc8a25116616c2bdf9e89654726818f6cc98ea49be7c237ff83f2515c8b8d0 0.0s
=> => exporting attestation manifest sha256:72de476726dfd1908a2eb2fa7fc474d934b44f8f542df28a0385e0aa7fbbf790 0.0s
=> => exporting manifest list sha256:2d0392d2dfbe657a70a82b6aa0586b08c3e103d14372b0f789fbb915a252ac69 149.6s
=> => naming to docker.io/library/tp-integrador-ayso-virtualizacion:latest 0.0s
=> => unpacking to docker.io/library/tp-integrador-ayso-virtualizacion:latest 0.2s
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/9ongugy5qovugorg04t9xorhp
```

Captura de la imagen de Docker en Docker Desktop



3. Y luego ejecutamos el contenedor con

```
docker run -it --rm tp-integrador-ayso-virtualizacion
```

```
ignac@nacho MINGW64 ~/Desktop/tp-integrador-ayso-virtualizacion (main)
$ docker run -it --rm tp-integrador-ayso-virtualizacion
Calculadora de IMC (Índice de Masa Corporal)
Ingresá tu peso en kilogramos (kg): 70
Ingresá tu altura en metros (m): 1.75

Tu IMC es: 22.86
Clasificación: Peso normal
```

4. Metodología Utilizada

Para este trabajo, primero investigamos conceptos básicos de Docker y contenedores, usando la documentación oficial y foros en línea. Luego, desarrollamos una calculadora de IMC en Python, que inicialmente usaba `input()` para pedir datos.

Al contenerizarla, detectamos que Docker no permite entrada interactiva sin flags especiales, por lo que adaptamos la ejecución usando `docker run -it` para habilitar la terminal.

Finalmente, creamos un Dockerfile para construir la imagen y realizamos pruebas para asegurar que la aplicación funcione correctamente en el contenedor.

Se utilizó Visual Studio Code para programar, Docker Desktop para contenerizar, Git para controlar versiones localmente y GitHub para alojar el código en la nube. El trabajo fue realizado siguiendo un proceso ordenado de diseño, prueba y corrección.

5. Resultados Obtenidos

¿Por qué usamos `-it`?

En un principio intentamos correr el contenedor así:

```
docker run --rm tp-integrador-ayso-virtualizacion
```

Pero ese comando nos tiró el siguiente mensaje de error

“EOFError: EOF when reading a line”

Esto pasó porque el script usa `input()` para pedir datos al usuario, pero **Docker por defecto no habilita la entrada interactiva** si no se lo indicamos explícitamente.

¿Qué hace `-it`?

- `-i (interactive)`: Mantiene abierta la entrada estándar (stdin), para que el contenedor pueda recibir datos del teclado.
- `-t (tty)`: Le da al contenedor acceso a una terminal virtual, como si fuera una consola real

Los resultados fueron muy positivos:

- Logramos una imagen funcional de Docker que corre nuestra app sin errores.
- Verificamos que el contenedor es liviano, rápido y portátil.
- El uso de Docker simplificó mucho el proceso de instalación y ejecución del programa

6.Conclusiones

Este trabajo permitió aplicar de forma práctica los conceptos de virtualización mediante contenedores, utilizando Docker como herramienta principal. Se logró contenerizar y ejecutar correctamente una aplicación en Python, comprendiendo el valor de los entornos aislados y portables.

Hoy en día, Docker es una tecnología clave en el desarrollo de software moderno, ya que permite simplificar el despliegue, facilitar las pruebas y asegurar que las aplicaciones funcionen de forma consistente en cualquier entorno. Aprender a usar Docker no solo refuerza el entendimiento de la virtualización, sino que también brinda una ventaja profesional importante para el trabajo en entornos reales de desarrollo y producción.

Este proyecto representa un primer paso sólido hacia el uso profesional de herramientas de contenerización, fundamentales en la industria actual.

Bibliografía

Listado de fuentes consultadas, utilizando normas básicas APA u otro formato consistente. Se sugiere incluir:

- Material de clase de la materia Arquitectura y Sistemas Operativos.

- Recursos en línea consultados para la implementación y comprensión de los conceptos de Docker como documentación oficial y artículos.
- OpenAI. (2025). *ChatGPT* [Modelo de lenguaje]. Disponible en <https://chat.openai.com/>

Anexo

- [Enlace a video en Youtube](#)
- [Enlace a repositorio en GitHub](#)

