

## Trabajo integrado – Arquitectura y Sistemas Operativos

# Tema: Virtualización

### Integrantes:

- Luna Gonzalo – gonzaloluna561@gmail.com
- González Federico – [fedeglz.ok@gmail.com](mailto:fedeglz.ok@gmail.com)

### Profesor:

- Ariel Enferrel

### Tutor:

- Rafael Ruiz

**Fecha de Entrega:** 05/06/2025

## **Índice:**

1. Introduccion
2. Marco teorico
3. Caso práctico
4. Metodologia utilizada
5. Resultados obtenidos
6. Conclusion
7. Biografia
8. Anexo

## **Introducción:**

En este trabajo integrador nos propusimos investigar y experimentar con tres tecnologías fundamentales en el campo de la virtualización: VirtualBox, Docker y Python, ya que permiten trabajar con entornos aislados y de maneras diferentes. La idea fue poner a prueba esas herramientas desde nuestro propio sistema operativo, Linux para Federico González y Windows para Gonzalo Luna, para poder entender mejor como funcionan, cuando conviene usar una u otra y que aportan al desarrollo de software y a la administración de sistemas.

## Marco Teórico:

Virtual Box: Es una herramienta que permite crear máquinas virtuales completas. Esto significa que dentro de una computadora podemos simular otra, con su sistema operativo, memoria, disco y red. Es útil para hacer pruebas sin arriesgar el sistema principal, aprender a usar otros sistemas operativos o simular redes de computadoras.

Docker: Trabaja con contenedores. Un contenedor aísla una aplicación y sus dependencias del resto del sistema. Son mucho más livianos que las máquinas virtuales y arrancan en segundos. Ideal para entornos de desarrollo moderno, microservicios y despliegue en la nube.

Python: Es un lenguaje de programación muy utilizado en el desarrollo de software moderno, tanto para el scripting como para automatización y desarrollo de aplicaciones web. En entornos virtuales o contenerizados, Python suele ser el lenguaje elegido por su facilidad, compatibilidad y versatilidad.

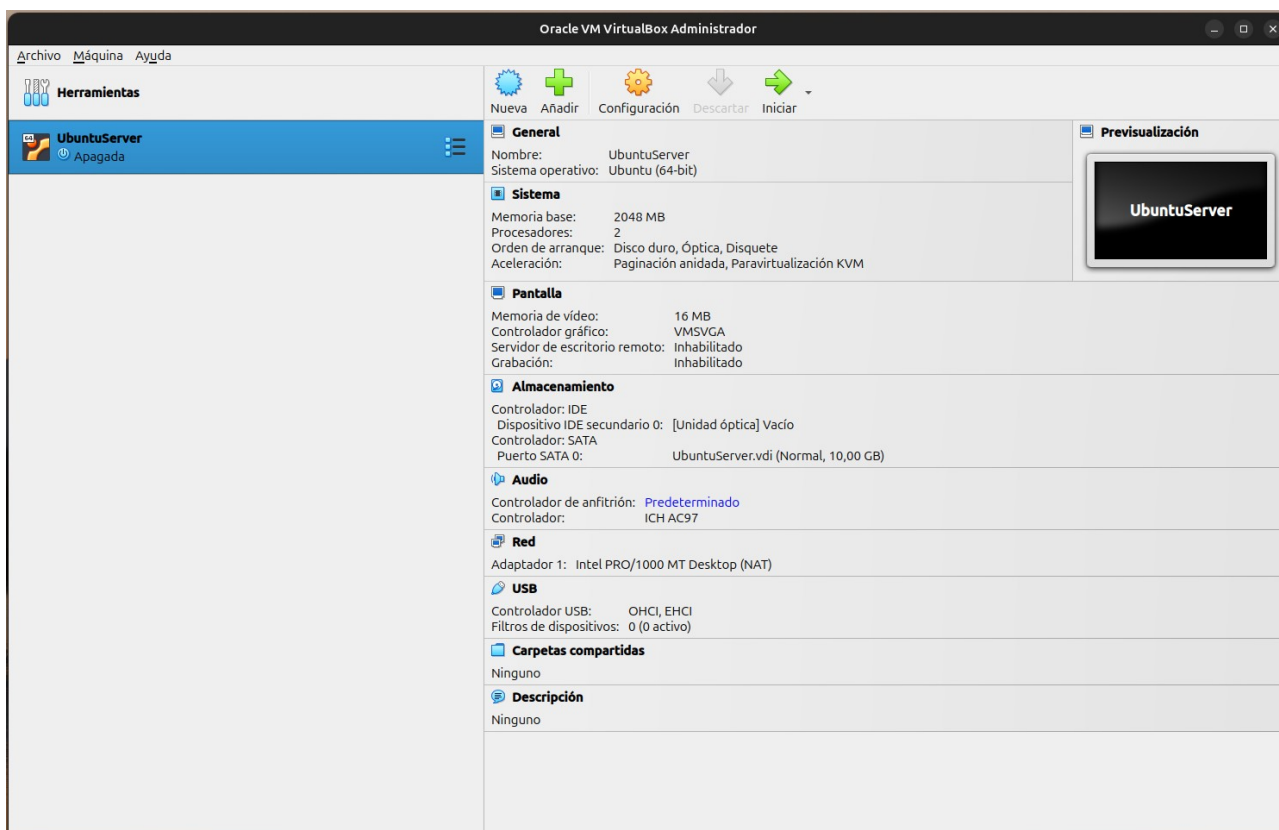
En nuestro trabajo, usamos Python dentro de una máquina virtual para verificar que el sistema estaba operativo y preparado para correr scripts. Esto simula lo que sucede en servidores reales, donde se ejecutan scripts de Python para automatizar tareas o levantar aplicaciones dentro de contenedores Docker

## Desarrollo del Trabajo

### Parte 1 – Virtual Box

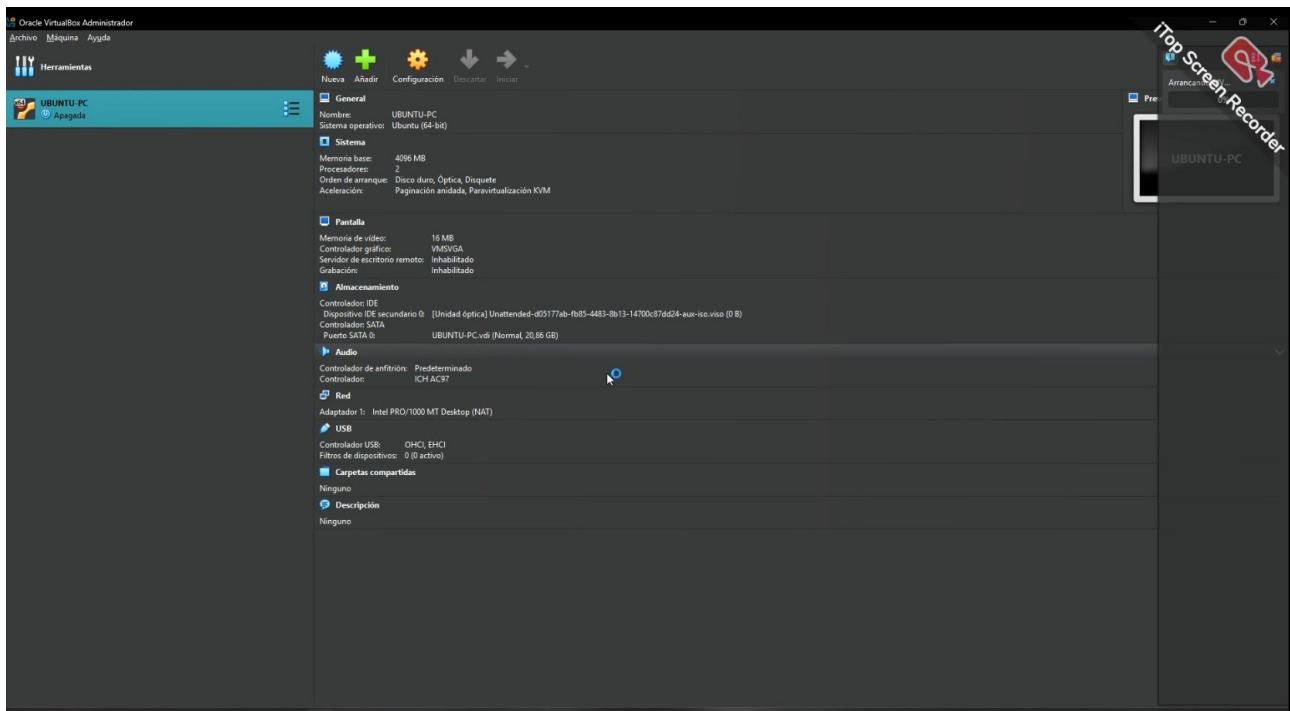
- Instalamos VirtualBox en ambos sistemas operativos. En Linux fue directo desde el repositorio oficial. En Windows, descargamos el instalador desde la web.
- Creamos una maquina virtual con Ubuntu Server. Le asignamos 2 gb de ram y 10GB de disco.
- Configuramos la red en modo puente para que se pueda acceder desde el sistema anfitrión.
- Sacamos capturas de cada paso para documentar el proceso

#### Linux:



## TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

### Windows:



### Parte 2 – Docker y Python

- En Linux, Federico González instaló Docker con apt y activo el servicio. En Windows Gonzalo Luna utilizó Docker Desktop que requiere WSL2.
- Probamos comandos básicos como: `docker run hello-world`.
- También instalamos Python dentro de la Máquina Virtual con Ubuntu Server. Verificamos la instalación con `python3 --version` y ejecutamos un pequeño script de prueba.

```
echo 'print("Esto es Python desde Ubuntu-Server en VM")' > prueba.py
```

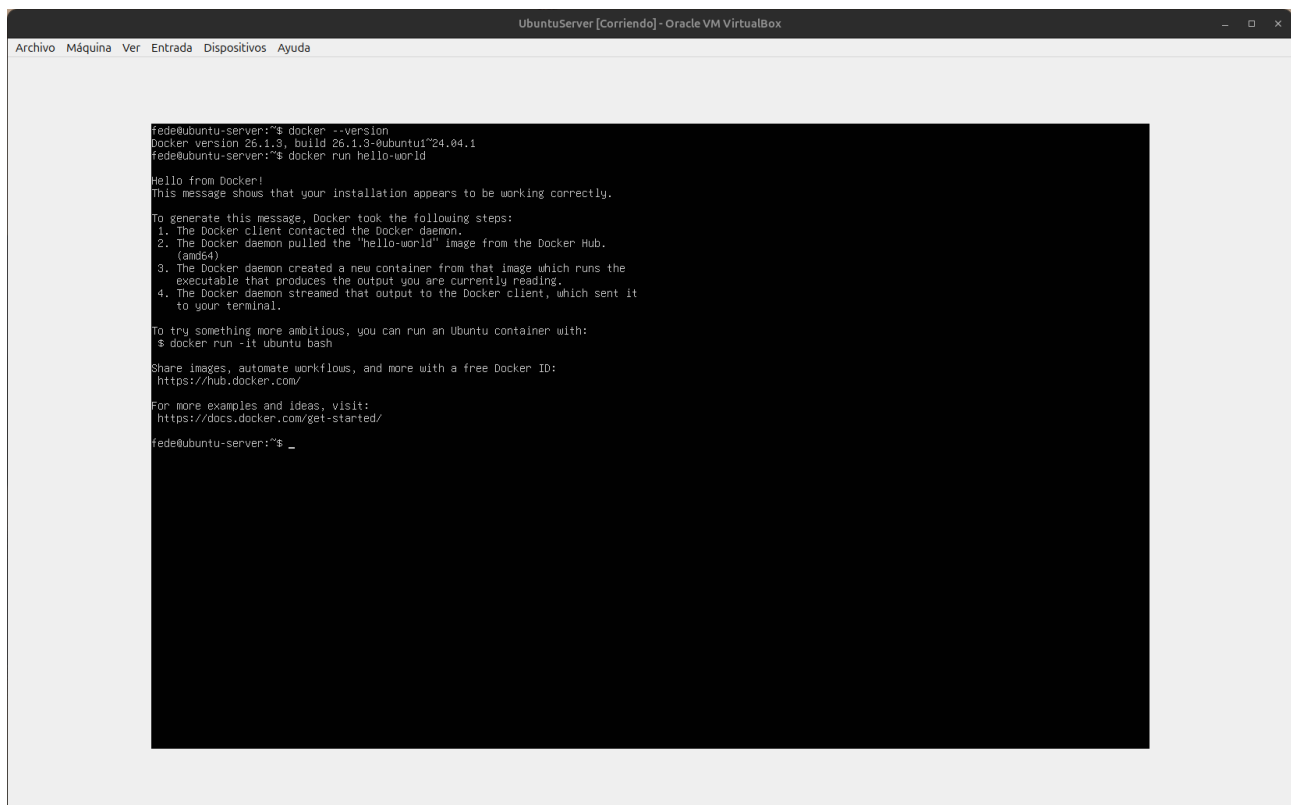
```
python3 prueba.py
```

```
Esto es Python desde Ubuntu-Server en VM
```

- Esto demuestra como se puede usar Python en entornos contenerizados o virtualizados, algo común en proyectos reales de automatización o desarrollo web.
- Agregamos capturas de la terminal, navegador y archivos usados.

## TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

### Linux:



```
UbuntuServer [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda

fed@ubuntu-server:~$ docker --version
Docker version 26.1.3, build 26.1.3-0ubuntu1~24.04.1
fed@ubuntu-server:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

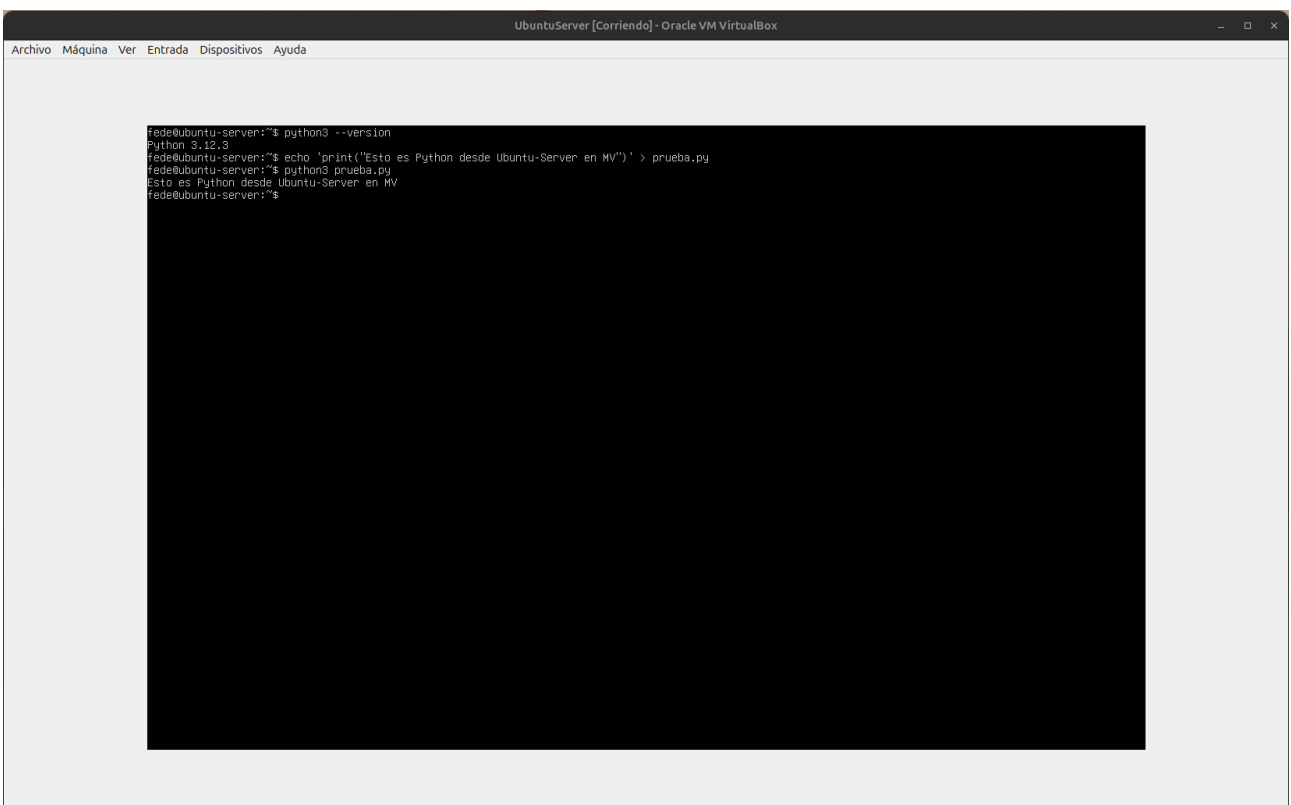
To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

fed@ubuntu-server:~$ _
```



```
UbuntuServer [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda

fed@ubuntu-server:~$ python3 --version
Python 3.12.3
fed@ubuntu-server:~$ echo 'print("Esto es Python desde Ubuntu-Server en MV")' > prueba.py
fed@ubuntu-server:~$ python3 prueba.py
Esto es Python desde Ubuntu-Server en MV
fed@ubuntu-server:~$
```

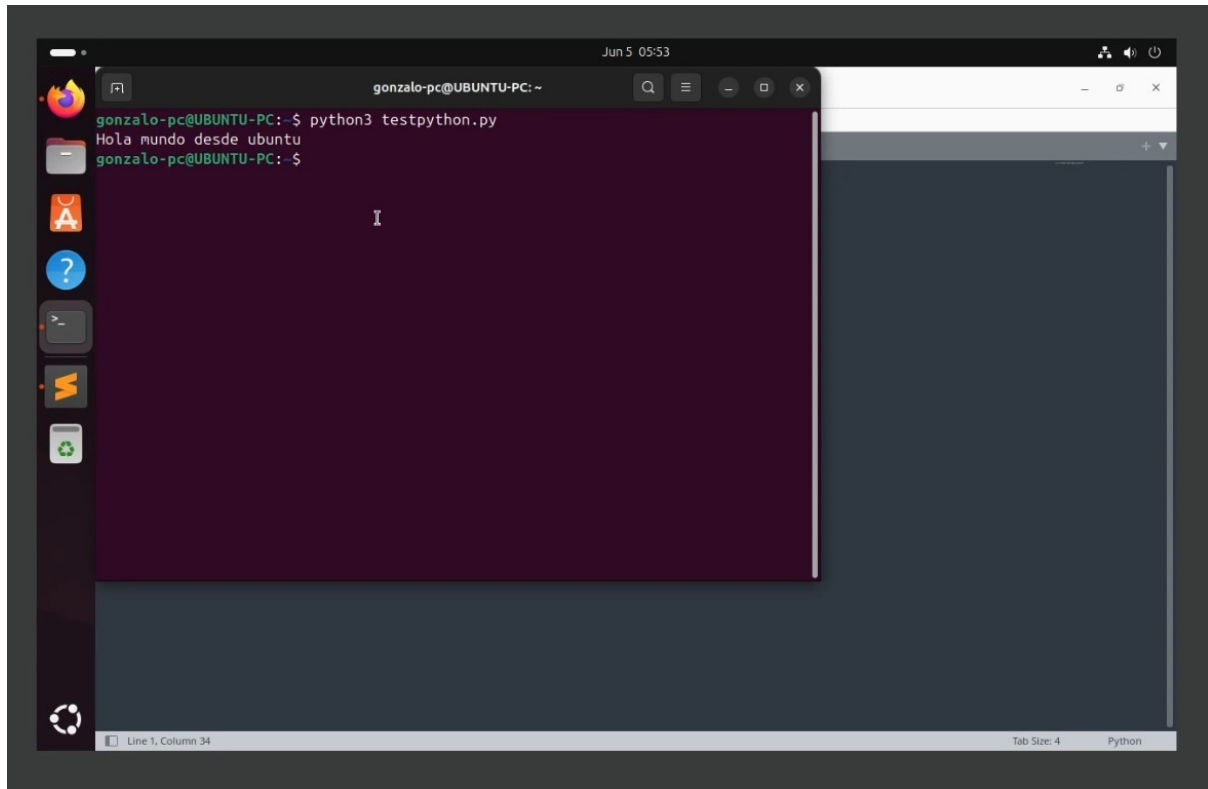
## TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

### Windows:

The screenshot shows a Windows desktop environment. On the left, a terminal window titled 'gonzalo-pc@UBUNTU-PC: ~' displays the output of the command 'sudo systemctl status docker'. The output indicates that the Docker service is loaded, enabled, and active (running) since Thursday, June 5, 2025, at 05:06:03 UTC. It also shows the service's main PID as 7218 (dockerd) and its CPU and memory usage. Below the status output, a list of log messages shows the service starting successfully. On the right, a web browser window displays the Docker installation guide for Ubuntu. The 'Table of contents' section is visible, listing prerequisites, installation methods, and next steps. The 'Install using the apt repository' method is highlighted.

The screenshot shows a Windows desktop environment. On the left, a terminal window titled 'gonzalo-pc@UBUNTU-PC: ~' displays the output of the command 'sudo apt-get install libstdc++-13-dev:amd64 binutils-x86-64-linux-gnu libpython3-dev:amd64 gcc-13-x86-64-linux-gnu g++-13-x86-64-linux-gnu'. The output shows the successful installation of these packages. Below the installation output, the terminal shows the command 'python3 --version' and its output 'Python 3.12.3'. On the right, a web browser window displays the course navigation page for 'TUPaD-AySO'. The 'Navegación' section is visible, listing various course components such as 'Página Principal', 'Área personal', 'Páginas del sitio', 'Mis cursos', 'TV', 'TUPaD-P1', 'Mate-25', 'TUPaD-AySO', 'Participantes', 'Competencias', 'Calificaciones', 'AySO - General', 'Introducción a la Arquitectura de Computadoras', 'Actividades', and 'Práctica'.

TECNICATURA UNIVERSITARIA EN  
PROGRAMACIÓN A DISTANCIA



The image shows a terminal window titled "gonzalo-pc@UBUNTU-PC: ~" with a search icon and window controls. The terminal output is as follows:

```
gonzalo-pc@UBUNTU-PC:~$ python3 testpython.py
Hola mundo desde ubuntu
gonzalo-pc@UBUNTU-PC:~$
```

The terminal has a dark purple background. A cursor is visible on the line following the prompt. To the right of the terminal is a web browser window, mostly obscured. The bottom status bar of the terminal shows "Line: 1, Column 34" on the left and "Tab Size: 4 Python" on the right.



### Parte 3 - Comparación:

- Armamos una tabla con los criterios: instalación, consumo de recursos, velocidad, facilidad de uso.
- Comentamos las diferencias que sentimos al usar ambas herramientas desde cada sistema operativo.
- Reflexionamos sobre cuándo conviene usar una u otra en el trabajo real.
- Además, destacamos cómo Python se integra con ambos entornos, y cómo su uso potencia el desarrollo dentro de contenedores o máquinas virtuales.

Criterio	VirtualBox	Docker	Python
Instalación	Manual en ambos SO	En Linux por apt, en Windows con Docker Desktop	Viene preinstalado en muchos distros; fácil de agregar
Consumo de recursos	Alto	Bajo	Muy bajo
Velocidad de ejecución	Lenta (como una PC real)	Rápida (segundos)	Inmediata (interprete ligero)
Facilidad de uso	Media (GUI completa)	Alta (CLI simple, herramientas modernas)	Muy alta (sintaxis clara, buena documentación)
Casos de uso	Simular entornos completos, redes	Despliegue ágil, microservicios	Automatización, scripting, desarrollo apps
Integración entre ellos	Se puede correr Docker y Python adentro	Python corre dentro de contenedores	Corre dentro de VM o contenedores sin problema
Portabilidad	Baja (depende del SO anfitrión)	Muy alta (contenedores reproducibles)	Alta (código multiplataforma)

## Conclusión

Hacer este trabajo nos ayudó a entender mejor cómo funcionan las máquinas virtuales y los contenedores. Con VirtualBox aprendimos a instalar un sistema operativo desde cero, configurar la red y manejar todo como si fuera una computadora aparte. Fue ideal para practicar cosas como Linux Server y entender bien cómo se arma un entorno completo.

Después, con Docker, vimos lo distinto que es trabajar con contenedores. Es mucho más rápido y liviano. Nos dimos cuenta de que es una herramienta muy útil para el día a día si trabajas en desarrollo o administración de sistemas.

Y también usamos Python dentro de la máquina virtual, para probar que podíamos correr scripts, y demostrar cómo se puede combinar con Docker. Esto nos mostró lo bien que se integran estas herramientas y lo importante que es saber usarlas juntas.

La verdad es que aprendimos un montón. No sólo a instalar y probar, sino a entender cuándo conviene usar una herramienta u otra. Todo esto nos va a servir mucho para los proyectos reales que vengan más adelante, o en la carrera.

## Anexos

- Comandos utilizados en Bash (instalación, uso de Docker, ejecución de scripts en Python).
- Capturas de pantalla de cada etapa del proceso, tanto en Linux como en Windows.
- Bibliografía: Documentación oficial de Docker, VirtualBox y Python; apuntes de cátedra; videos y material compartido por la UTN.
- **Link del video en YouTube:** <https://www.youtube.com/watch?v=Vdn5e6RzkVg>