



TRABAJO PRACTICO INTEGRADOR VIRTUALIZACION

ALUMNOS: FABIAN IGNACIO CARDOSO – fabian21cf@gmail.com;
LAUTARO ARIEL CEJAS – laut217@live.com

MATERIA: ARQUITECTURA Y SISTEMAS OPERATIVOS

PROFESOR/A: DIEGO LOBOS

FECHA DE ENTREGA: 05/06/2025

ÍNDICE

Trabajo Practico Integrador VIRTUALIZACION

1. Introducción	Pág. 3
2. Marco Teórico.....	Pág. 4
2.1. Virtualización.....	Pág. 4-12
2.2. Concepto.....	Pág. 4
2.3. Tipos de Virtualización.....	Pág. 5-7
2.4. Hipervisores.....	Pág. 7-9
2.5. Beneficios y Desafíos.....	Pág. 10-11
2.6. Imágenes ISO y Snapshots.....	Pág. 12
2.7. Redes Virtuales	Pág. 12
3. Caso Practico Virtualización.....	Pág. 13-24
4. Conclusión	Pág. 25
5. Referencias Bibliográficas.....	Pág. 26
6. Anexo.....	Pág. 26



INTRODUCCIÓN.

Elegimos este tema por su relevancia en el campo de la programación, ya que nos permite explorar cómo los avances tecnológicos, como la virtualización, pueden transformar las prácticas empresariales y técnicas relacionadas con la administración de recursos computacionales. Además, este tema tiene una gran importancia en la formación de un técnico en programación porque nos proporciona una comprensión profunda de herramientas y tecnologías que son esenciales para la industria actual. La virtualización, por ejemplo, es una tecnología clave en la reducción de costos, la mejora en la gestión de sistemas y la creación de entornos flexibles para el desarrollo de software. Aprender sobre estas herramientas prepara al técnico en programación para enfrentarse a retos complejos, como la implementación de soluciones eficientes y la resolución de problemas relacionados con la compatibilidad entre sistemas operativos, aplicaciones y hardware.

Con este trabajo, tenemos el propósito de poder desarrollar competencias analíticas y técnicas que nos permitan implementar soluciones innovadoras y eficientes en proyectos de programación. Además, se busca poder identificar y aplicar tecnologías como la virtualización para mejorar procesos empresariales y resolver problemas prácticos. Entre los objetivos específicos se encuentra el fortalecimiento de habilidades críticas para evaluar el desempeño de diferentes sistemas, así como la capacidad de integrar nuevas tecnologías en entornos dinámicos y diversos.

2. Marco Teórico

2.1 VIRTUALIZACIÓN.

La virtualización representa una de las tecnologías más transformadoras en el ámbito de los sistemas computacionales modernos. Según Tanenbaum y Bos (2021), la virtualización permite la creación de versiones virtuales de recursos físicos, incluyendo hardware, sistemas operativos, dispositivos de almacenamiento y recursos de red. Esta tecnología fundamental ha revolucionado la manera en que se diseñan, implementan y gestionan los sistemas informáticos contemporáneos.

2.2 Concepto de Virtualización.

La virtualización es el proceso mediante el cual se abstraen los recursos físicos de una máquina para crear entornos virtuales independientes. Esta tecnología permite la ejecución de múltiples sistemas operativos en un solo equipo, optimizando el uso de hardware y mejorando la eficiencia operativa. Stallings (2020) señala que la virtualización ha sido clave en la evolución de los sistemas operativos modernos, permitiendo la separación de aplicaciones y sistemas mediante entornos seguros y aislados. Estableciendo que la virtualización es el proceso de presentar un conjunto de recursos computacionales de tal manera que pueden ser accedidos de forma que no esté restringida por la configuración física o la ubicación geográfica. Esta definición abarca no solo la virtualización de hardware, sino también la virtualización de aplicaciones, redes y almacenamiento.

La virtualización, según Coulouris, Dollimore y Kindberg (2019), se fundamenta en el principio de abstracción, donde se crea una capa intermedia entre el hardware físico y el software que se ejecuta sobre él. Esta capa de abstracción,

conocida como hipervisor o monitor de máquina virtual, gestiona y controla el acceso a los recursos físicos.

2.3 Tipos de Virtualización.

La virtualización se clasifica en varias categorías, entre ellas:

- **Virtualización de servidores:** Permite la ejecución de múltiples sistemas operativos en un solo servidor físico (Silberschatz, Galvin, & Gagne, 2018).
- **Virtualización de aplicaciones:** Facilita la ejecución de programas en entornos aislados sin afectar el sistema operativo base (Coulouris, Dollimore, & Kindberg, 2019).
- **Virtualización de almacenamiento:** Optimiza la administración de datos mediante unidades virtuales que mejoran la accesibilidad y seguridad (Deitel & Deitel, 2019).
- **Virtualización de red:** Simula redes físicas para mejorar la conectividad y seguridad, permitiendo la creación de entornos distribuidos eficientes (Silberschatz et al., 2018).

Características Principales de la Virtualización:

- Abstracción del Hardware.

La virtualización permite la separación entre el hardware físico y los sistemas operativos que se ejecutan sobre él. Según Tanenbaum y Bos (2021), esta abstracción es clave para mejorar la eficiencia y flexibilidad en la administración de recursos, ya que permite que múltiples sistemas operativos compartan el mismo hardware sin interferencias.

- Aislamiento de Máquinas Virtuales.

Cada máquina virtual (VM) opera de manera independiente, sin afectar a las demás. Stallings (2020) explica que este aislamiento es fundamental para la

seguridad y estabilidad del sistema, ya que evita que fallos en una VM afecten a otras. Además, permite la ejecución de múltiples entornos sin comprometer el rendimiento general.

- Gestión Dinámica de Recursos.

La virtualización optimiza el uso de CPU, memoria y almacenamiento mediante la asignación dinámica de recursos. Silberschatz, Galvin y Gagne (2018) destacan que los hipervisores modernos pueden ajustar automáticamente la cantidad de recursos asignados a cada VM según la demanda, mejorando la eficiencia operativa.

- Compatibilidad con Múltiples Sistemas Operativos.

Una de las ventajas más importantes de la virtualización es la capacidad de ejecutar diferentes sistemas operativos en un mismo hardware. Según Coulouris, Dollimore y Kindberg (2019), esto permite a los administradores de TI probar y desarrollar aplicaciones en distintos entornos sin necesidad de hardware adicional.

- Seguridad y Control de Acceso

La virtualización mejora la seguridad al permitir la creación de entornos aislados para pruebas y desarrollo. Deitel y Deitel (2019) explican que los hipervisores incluyen mecanismos de control de acceso que restringen la interacción entre máquinas virtuales, reduciendo el riesgo de ataques y vulnerabilidades.

- Escalabilidad y Flexibilidad

La capacidad de escalar recursos de manera rápida es una de las razones por las que la virtualización es ampliamente utilizada en centros de datos y computación en la nube. Stallings (2020) señala que los entornos virtualizados pueden expandirse o reducirse según las necesidades del usuario, sin requerir cambios físicos en el hardware.

- Reducción de Costos Operativos

La consolidación de servidores mediante virtualización reduce significativamente los costos de hardware y mantenimiento. Tanenbaum y Bos (2021) explican que, al ejecutar múltiples sistemas en un solo servidor físico, las empresas pueden disminuir el consumo energético y los gastos asociados a la infraestructura de TI.

- Virtualización de Red y Almacenamiento

Además de los sistemas operativos, la virtualización se extiende a redes y almacenamiento. Silberschatz et al. (2018) describen cómo las redes virtualizadas permiten la creación de entornos de comunicación seguros y eficientes, mientras que la virtualización de almacenamiento optimiza la gestión de datos mediante unidades virtuales.

Componentes Fundamentales.

Deitel y Deitel (2019) identifican los componentes esenciales de un sistema virtualizado:

Hipervisor: También denominado Virtual Machine Monitor (VMM), constituye la capa de software que gestiona una o más máquinas virtuales. El hipervisor es responsable de la asignación de recursos físicos a las máquinas virtuales y de mantener el aislamiento entre ellas.

Máquina Virtual (Virtual Machine): Representa una abstracción completa de un sistema computacional que incluye procesador virtual, memoria virtual, dispositivos de entrada/salidas y almacenamiento virtuales.

Sistema Operativo Huésped (Guest Operating System): Es el sistema operativo que se ejecuta dentro de una máquina virtual, sin conocimiento de que está siendo virtualizado.

Sistema Operativo Anfitrión (Host Operating System): En ciertos tipos de virtualización, representa el sistema operativo que se ejecuta directamente sobre el hardware físico y sobre el cual se instala el hipervisor.

2.4 Hipervisores.

El hipervisor es el componente clave en la virtualización de sistemas operativos, ya que permite la ejecución de múltiples máquinas virtuales sobre un mismo hardware físico. Según Tanenbaum y Bos (2021), los hipervisores actúan como una capa de abstracción entre el hardware y los sistemas operativos invitados, gestionando los recursos y asegurando la eficiencia en la ejecución de procesos virtualizados.

Definición y Función del Hipervisor.

Un hipervisor, también conocido como monitor de máquina virtual (VMM), es un software o firmware que permite la creación y administración de máquinas virtuales (VMs). Stallings (2020) explica que el hipervisor asigna dinámicamente los recursos del hardware físico a las máquinas virtuales, permitiendo que cada una opere de manera independiente.

Tipos de Hipervisores:

Los hipervisores se clasifican en dos grandes categorías, según Stallings (2020) y Coulouris, Dollimore y Kindberg (2019):

- **Hipervisores Tipo 1 (Bare-Metal).**

Los hipervisores tipo 1 se ejecutan directamente sobre el hardware físico, sin necesidad de un sistema operativo anfitrión. Son utilizados en entornos empresariales y centros de datos debido a su alto rendimiento y seguridad. Ejemplos destacados incluyen:

- VMware ESXi

- Microsoft Hyper-V

- Xen

Ventajas:

- Mayor eficiencia: Acceso directo al hardware sin intermediarios.

- Menor sobrecarga: No depende de un sistema operativo anfitrión.

- Mayor seguridad: Aislamiento completo de las máquinas virtuales.

- Hipervisores Tipo 2 (Hosted).

Los hipervisores tipo 2 se ejecutan sobre un sistema operativo anfitrión, lo que los hace más accesibles para entornos de desarrollo y pruebas. Ejemplos incluyen:

- Oracle VirtualBox

- VMware Workstation

- Parallels Desktop

Ventajas:

- Facilidad de instalación: Se ejecutan sobre un sistema operativo existente.

- Compatibilidad con hardware estándar: No requieren configuraciones avanzadas.

- Ideal para pruebas y desarrollo: Permiten ejecutar múltiples entornos en una misma máquina.

Arquitectura y Funcionamiento.

Según Tanenbaum y Bos (2021), la arquitectura de un hipervisor se basa en la gestión de recursos físicos y virtuales. Los hipervisores tipo 1 interactúan directamente con el hardware, mientras que los tipos 2 dependen del sistema operativo anfitrión.

Módulos Principales:

Silberschatz et al. (2018) identifican los siguientes módulos clave en la arquitectura de un hipervisor:

- Gestión de CPU: Asigna ciclos de procesamiento a cada VM.
- Administración de memoria: Distribuye RAM de manera eficiente.
- Control de almacenamiento: Maneja discos virtuales y acceso a datos.
- Virtualización de red: Facilita la comunicación entre máquinas virtuales.

Los hipervisores representan una tecnología esencial en la virtualización de sistemas operativos, permitiendo la ejecución eficiente de múltiples entornos sobre un mismo hardware. Su correcta implementación optimiza recursos, mejora la seguridad y facilita la administración de sistemas distribuidos.

2.5 Beneficios y Desafíos.

Beneficios

- **Optimización de recursos:** Reduce el número de servidores físicos requeridos y mejora la eficiencia energética (Deitel & Deitel, 2019).
- **Escalabilidad:** Permite crear y administrar máquinas virtuales rápidamente en entornos empresariales (Silberschatz et al., 2018).

- **Seguridad:** Facilita la creación de entornos aislados para pruebas y desarrollo, reduciendo riesgos asociados a ataques cibernéticos (Coulouris et al., 2019).
- **Desafíos**
- **Consumo de recursos:** La virtualización puede generar una sobrecarga en el hardware si no se administra adecuadamente (Stallings, 2020).
- **Compatibilidad:** No todos los sistemas operativos funcionan de manera óptima en entornos virtuales, lo que puede afectar el rendimiento de ciertas aplicaciones (Tanenbaum & Bos, 2021).

Aplicaciones de la Virtualización

La virtualización es utilizada en **centros de datos, infraestructura de nube, desarrollo de software y seguridad informática**. En el ámbito académico, se emplea para crear entornos de prueba sin riesgos para el sistema físico (Silberschatz et al., 2018).

Virtualización y Computación en la Nube

La virtualización es un pilar fundamental de la computación en la nube. Según Coulouris et al. (2019), la virtualización permite la creación de entornos escalables y flexibles para la gestión de recursos en la nube. Los principales modelos de computación en la nube incluyen:

- **Infraestructura como Servicio (IaaS):** Provisión de recursos virtualizados como servidores y almacenamiento.
- **Plataforma como Servicio (PaaS):** Entornos de desarrollo y ejecución de aplicaciones.
- **Software como Servicio (SaaS):** Aplicaciones accesibles a través de internet sin necesidad de instalación local.

2.6 Imágenes ISO y Snapshots.

Las imágenes ISO son archivos que contienen una copia exacta de un sistema de archivos, comúnmente utilizadas para la instalación de sistemas operativos. Según Deitel & Deitel (2019), los snapshots permiten guardar el estado de una máquina virtual en un momento determinado, facilitando la recuperación ante fallos o pruebas de software.

2.7 Redes Virtuales

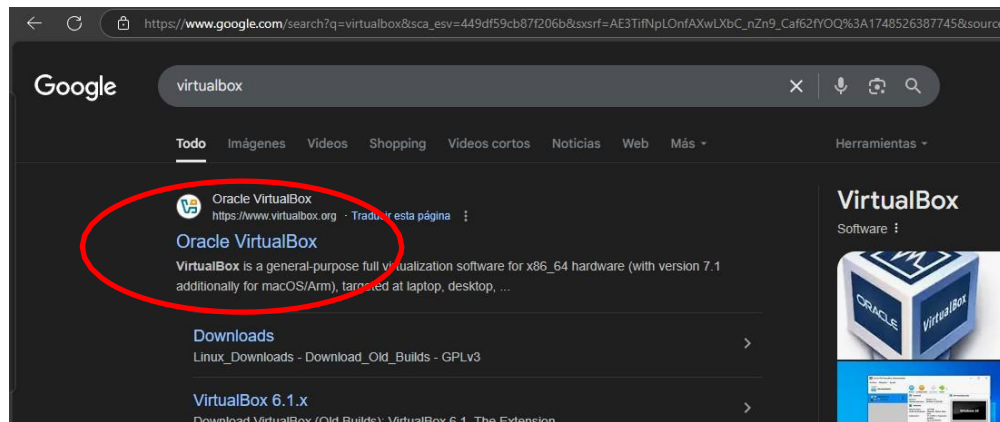
Las redes virtuales permiten la comunicación entre máquinas virtuales y el host. Se dividen en:

- **NAT (Network Address Translation):** La máquina virtual accede a Internet usando la IP del host. Ideal para entornos donde se requiere conectividad externa sin configuración avanzada (Coulouris et al., 2019).
- **Bridge:** La VM obtiene una dirección IP en la misma red que el host, permitiendo comunicación directa con otros dispositivos de la red física (Silberschatz et al., 2018).
- **Host-only:** La comunicación está limitada al host y otras máquinas virtuales en la misma red privada, sin acceso a Internet (Stallings, 2023).

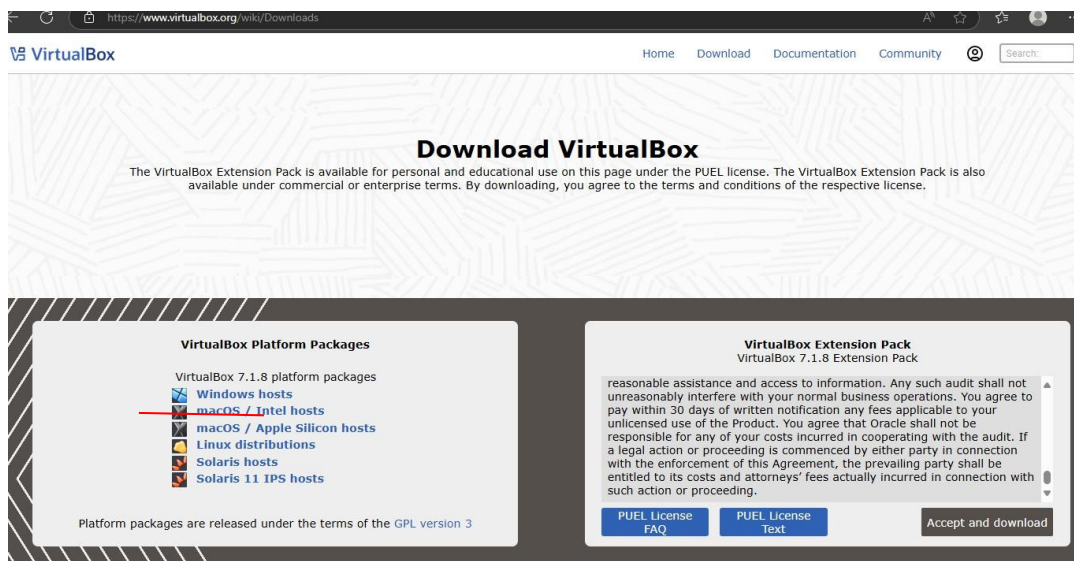
3. Caso Practico de Virtualización.

Después de haber visto los fundamentos teóricos de la virtualización (Hipervisor tipo 2) vamos a implementar un entorno virtual utilizando VirtualBox para instalar y configurar Linux, aplicando los conceptos vistos.

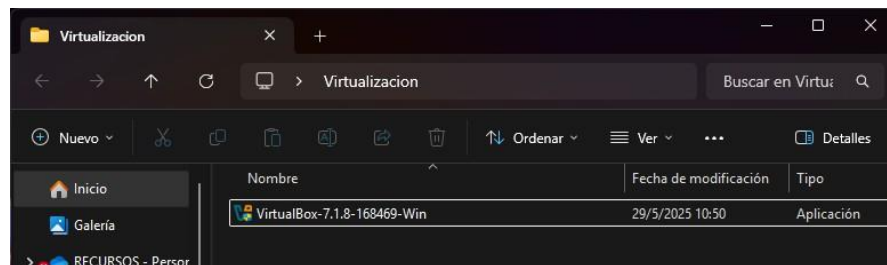
- Primero vamos a entrar a nuestro navegador y buscaremos VirtualBox.



- Según nuestro sistema operativo utilizado, lo seleccionaremos y descargaremos. En nuestro caso seleccionaremos Windows host.

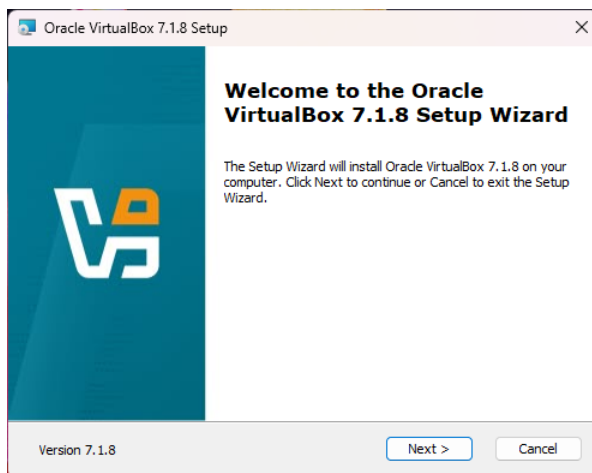


- Una vez seleccionado nuestro sistema operativo de origen, se procederá a descargar el instalador del programa y haremos doble clic sobre el mismo.

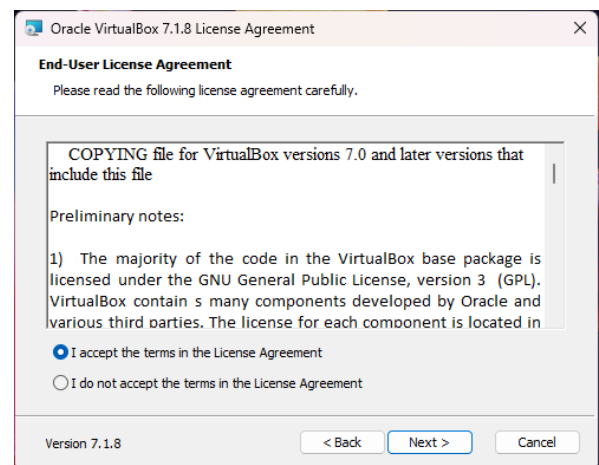


- A continuación, se lanza el instalador y empieza la configuración de instalación del programa.

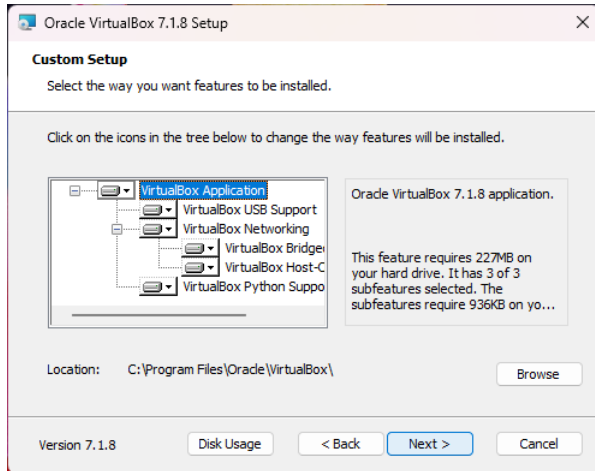
1 - Inicio de instalación



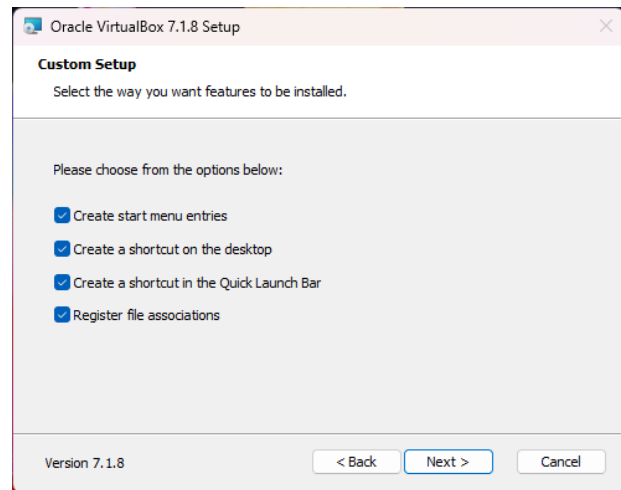
2 - Aceptar Terminos y Condiciones



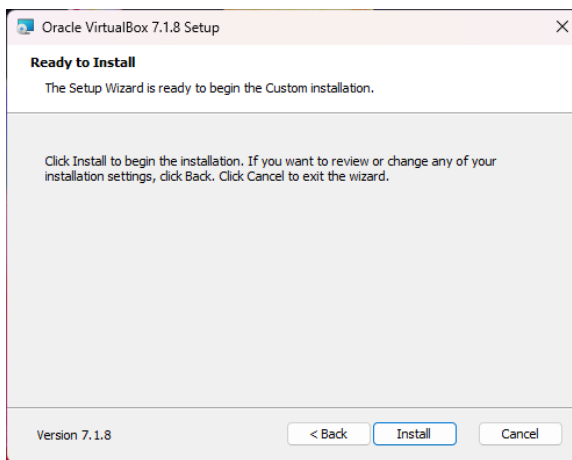
3 - Selección de ubicación del Programa



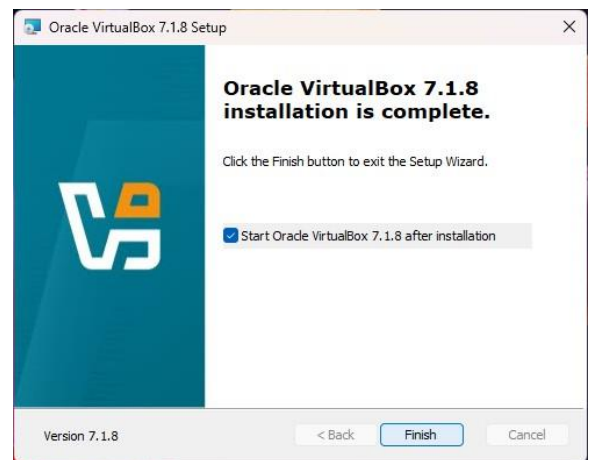
4- Selección de preferencias de instalación



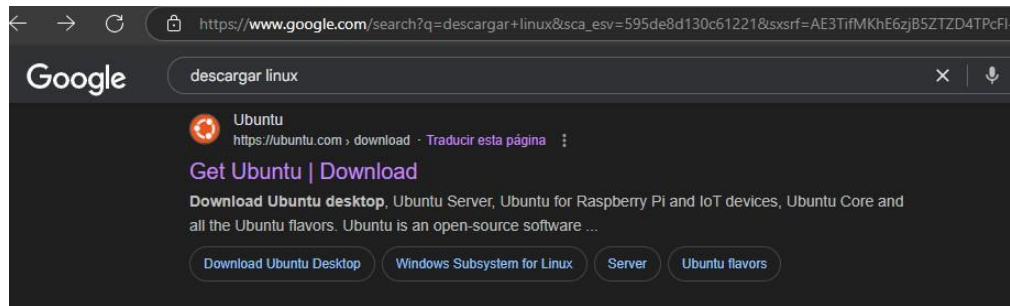
5 - Se Procede a Instala el Programa



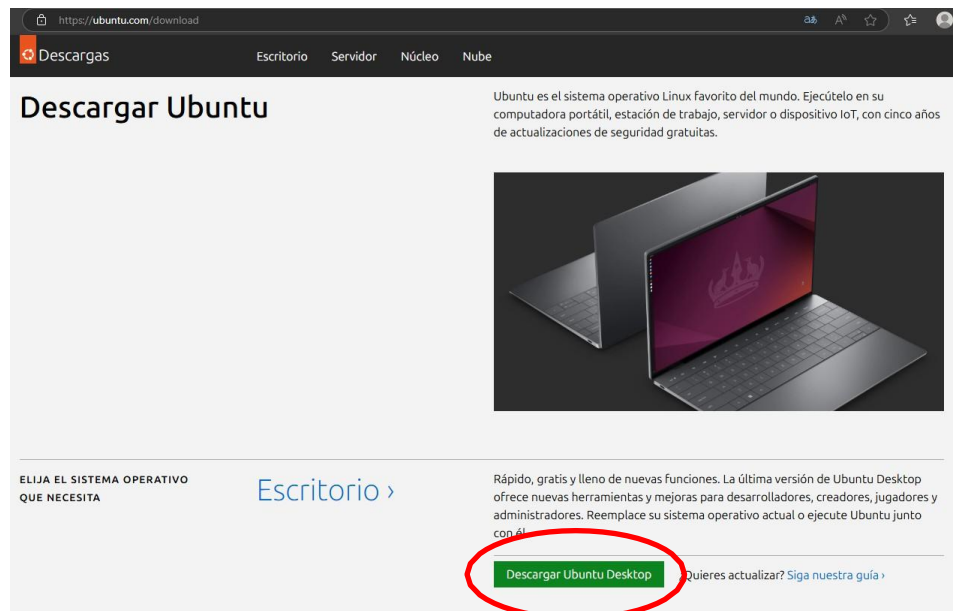
6- Finaliza la instalación



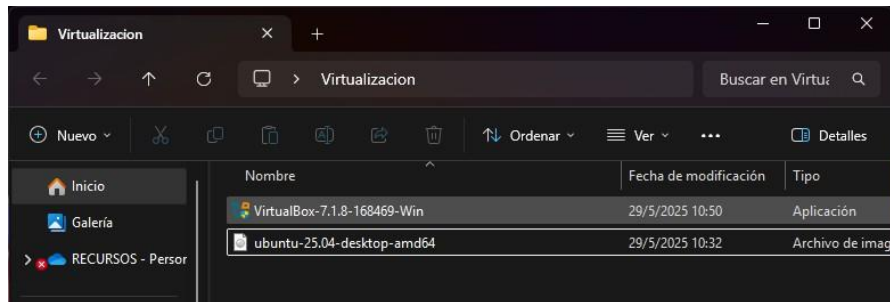
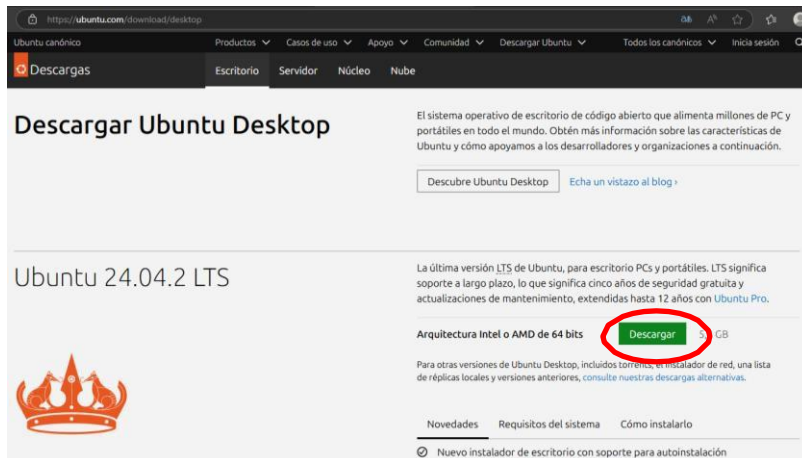
7 - Descarga de ISO de Linux



- Ubuntu es una distribución de Linux basada en Debian, diseñada para ser fácil de usar y accesible tanto para principiantes como para usuarios avanzados. Cuando descargas Ubuntu, obtienes un sistema operativo basado en Linux listo para instalar. Se ofrece en diferentes versiones, como Ubuntu Desktop (para computadoras personales) y Ubuntu Server (para servidores). Además, hay variantes oficiales con diferentes entornos gráficos, como Kubuntu (con KDE) o Xubuntu (con XFCE)



- Seleccionamos la versión para de Ubuntu que deseamos descargar.



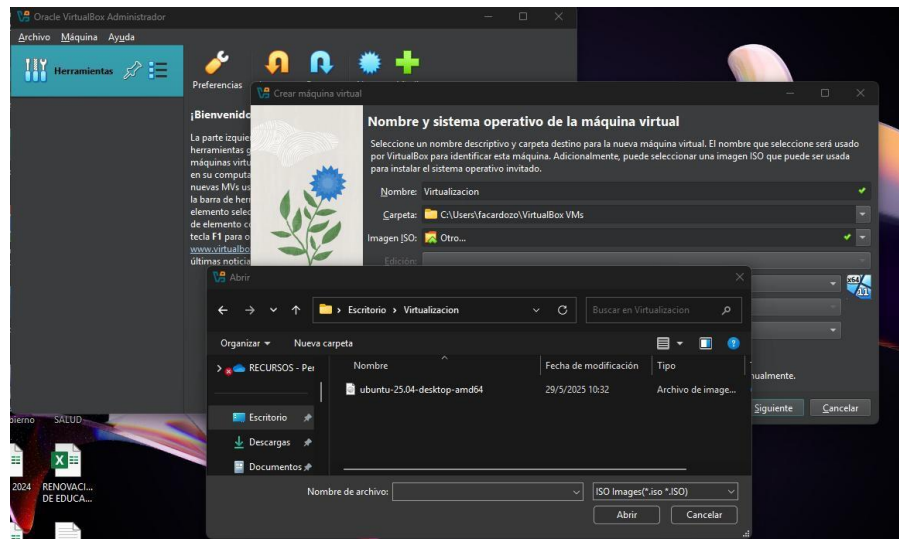
- Una vez descargada la imagen ISO de Linux, ejecutamos el programa VirtualBox y seleccionaremos la opción “Nueva” (se creará una máquina virtual).



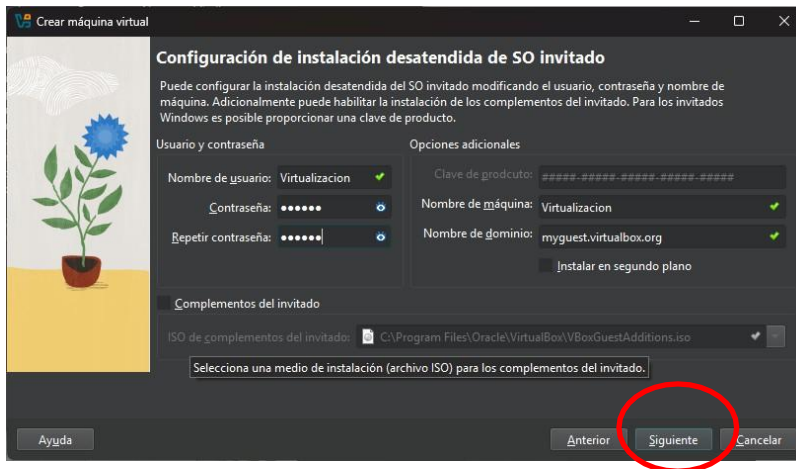
- Aquí deberemos montar la imagen ISO.



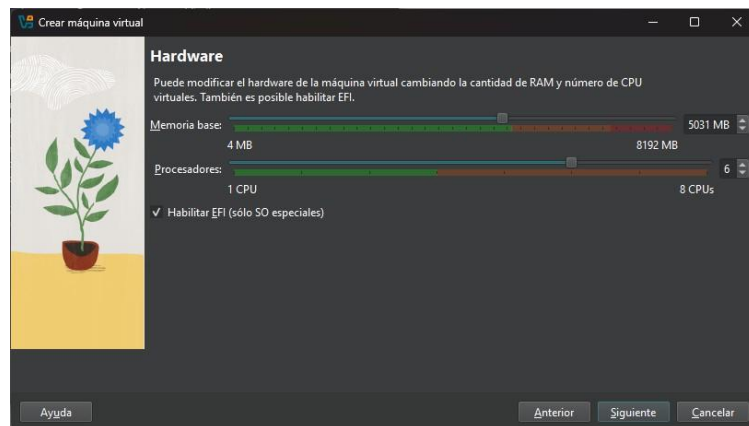
- Se nos abrirá una ventana en la cual deberemos buscar en nuestra carpeta de descargas la ISO.



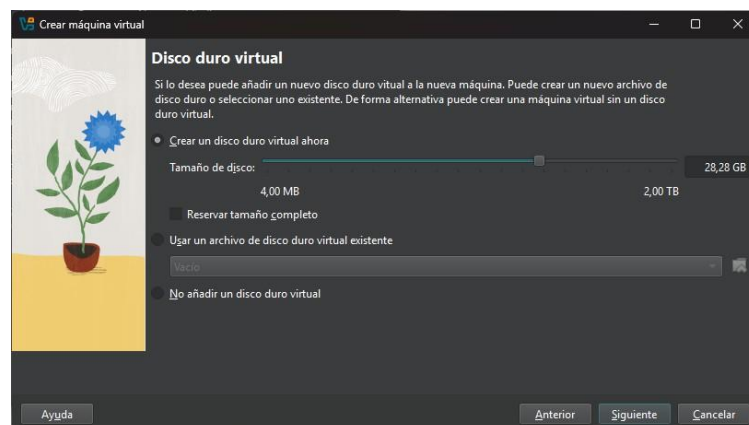
- Elegimos un nombre de usuario y contraseña para más seguridad.



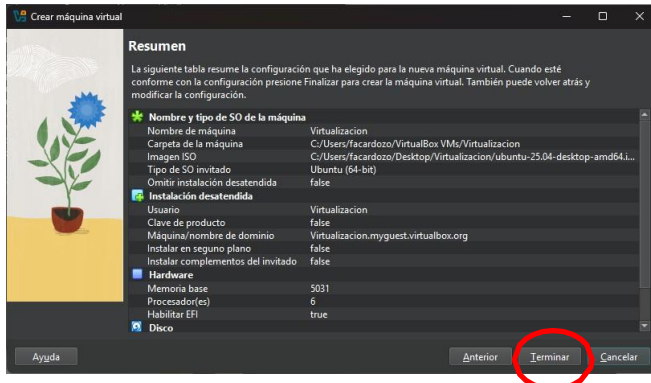
- Ahora según los recursos físicos de nuestro Computador nos dará la opción de seleccionar, la cantidad de memoria RAM y cantidad del procesador que vamos a virtualizar.



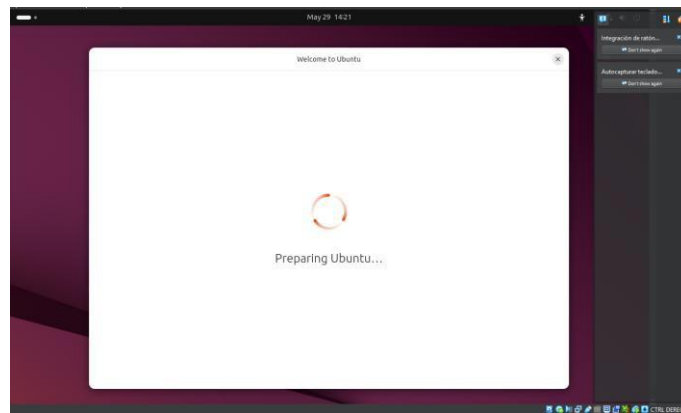
- Seguido, nos va a pedir que elijamos el tamaño del Disco que le vamos a asignar a esta máquina virtual.



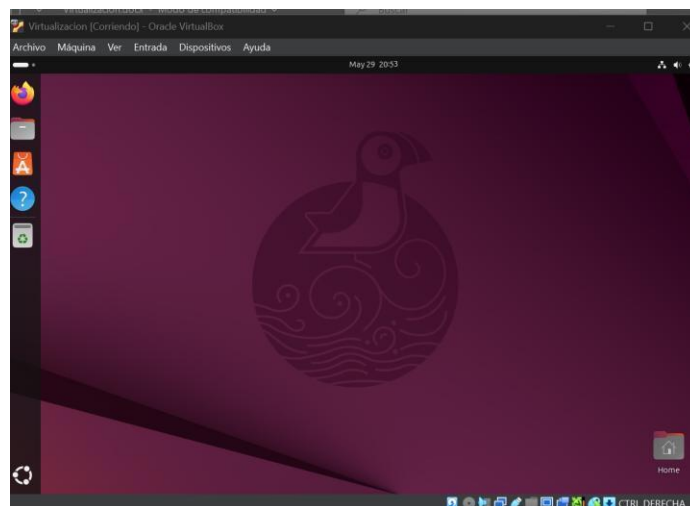
- Habiendo configurado todas las opciones presionamos en “Terminar”.



- Se procede con la instalación y configuración del sistema.

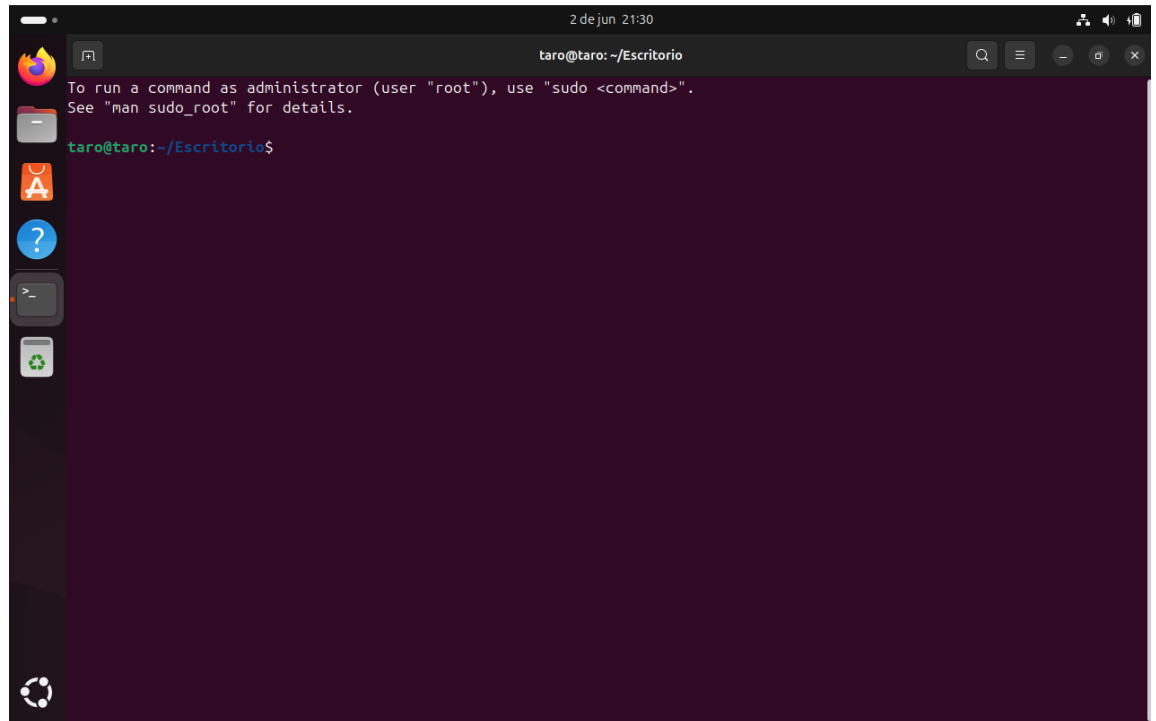


- Ahora Linux está listo para usarse.



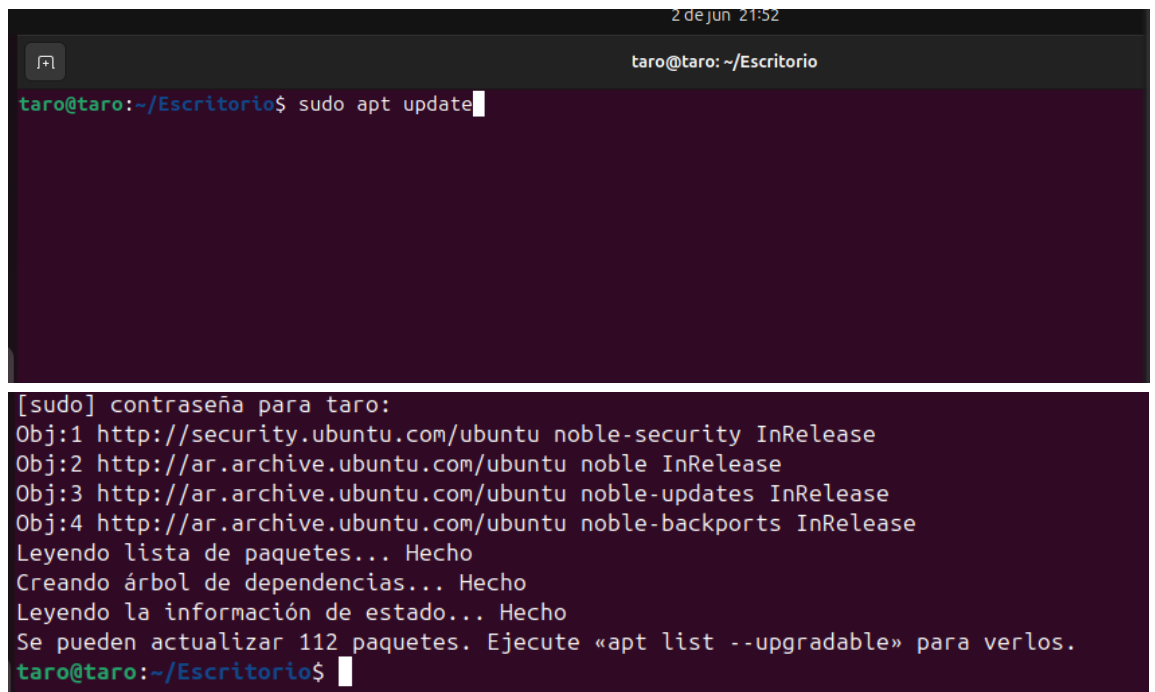
Ahora que tenemos Ubuntu funcionando efectivamente podremos instalar Python

- Continuaremos con la instalación de Python. Para ello lo primero que debemos hacer es abrir la terminal.

A terminal window titled 'taro@taro: ~/Escritorio' is open on an Ubuntu desktop. The window shows a message about running commands as administrator: 'To run a command as administrator (user "root"), use "sudo <command>". See "man sudo_root" for details.' Below this, the prompt 'taro@taro:~/Escritorio\$' is visible. The desktop background is dark, and the terminal window has a dark theme with a purple border.

```
taro@taro:~/Escritorio$
```

- Escribimos “sudo apt update”. Este comando actualizará el catálogo de paquetes para traernos las versiones más nuevas.

A terminal window titled 'taro@taro: ~/Escritorio' is open. The command 'sudo apt update' has been entered. The output shows the progress of the update process, including checking for updates from various sources and calculating the upgrade. The prompt 'taro@taro:~/Escritorio\$' is visible at the bottom.

```
taro@taro:~/Escritorio$ sudo apt update
[sudo] contraseña para taro:
Obj:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Obj:2 http://ar.archive.ubuntu.com/ubuntu noble InRelease
Obj:3 http://ar.archive.ubuntu.com/ubuntu noble-updates InRelease
Obj:4 http://ar.archive.ubuntu.com/ubuntu noble-backports InRelease
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se pueden actualizar 112 paquetes. Ejecute «apt list --upgradable» para verlos.
taro@taro:~/Escritorio$
```

- Una vez que tenemos el catálogo actualizado procedemos con la instalación del software. Ejecutamos el script “sudo apt install python3”.

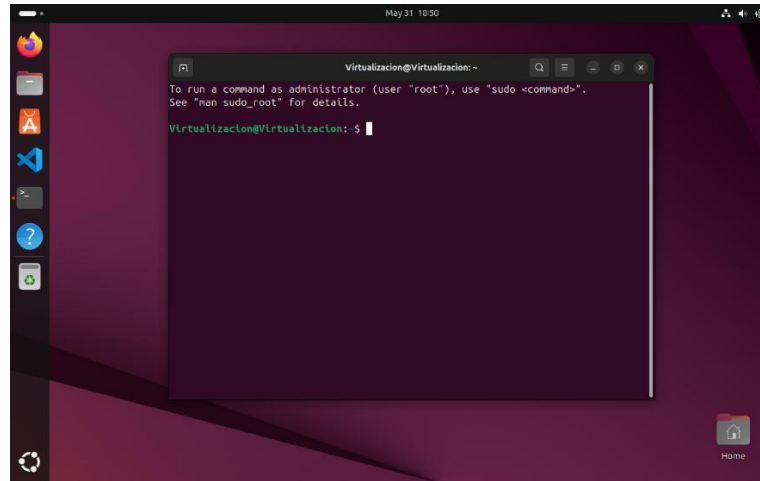
```
taro@taro:~/Escritorio$ sudo apt install python3
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
python3 ya está en su versión más reciente (3.12.3-0ubuntu2).
Fijado python3 como instalado manualmente.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 112 no actualizados.
taro@taro:~/Escritorio$ S
```

- Listo. Para corroborar la correcta instalación de este escribiremos “python3 --version”. El mismo nos dirá qué versión hemos instalado en nuestro sistema.

```
taro@taro:~/Escritorio$ python3 --version
Python 3.12.3
taro@taro:~/Escritorio$
```

Con Python instalado en nuestro sistema Ubuntu podremos crear el programa que queramos codificado en, valga la redundancia, Python.

- Vamos a abrir la Terminal de comando.



- Con el siguiente script crearemos una carpeta “mkdir Python”

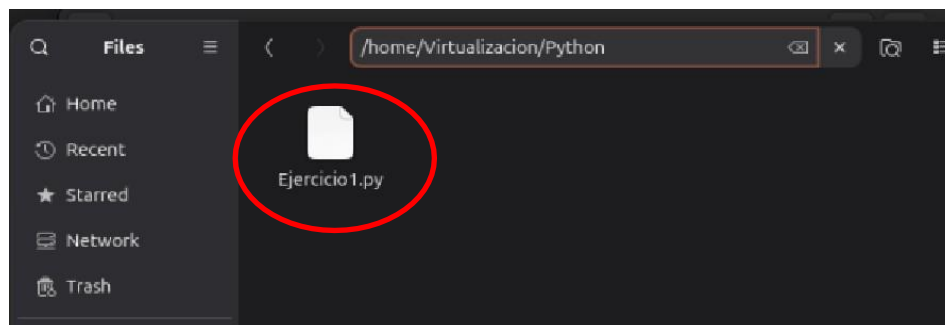
```
Virtualizacion@Virtualizacion: ~/Python
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Virtualizacion@Virtualizacion:~$ #!/bin/bash
Virtualizacion@Virtualizacion:~$ mkdir Python
Virtualizacion@Virtualizacion:~$ cd /home/Virtualizacion/Python
Virtualizacion@Virtualizacion:~/Python$ touch Ejercicio1.py
Virtualizacion@Virtualizacion:~/Python$
```

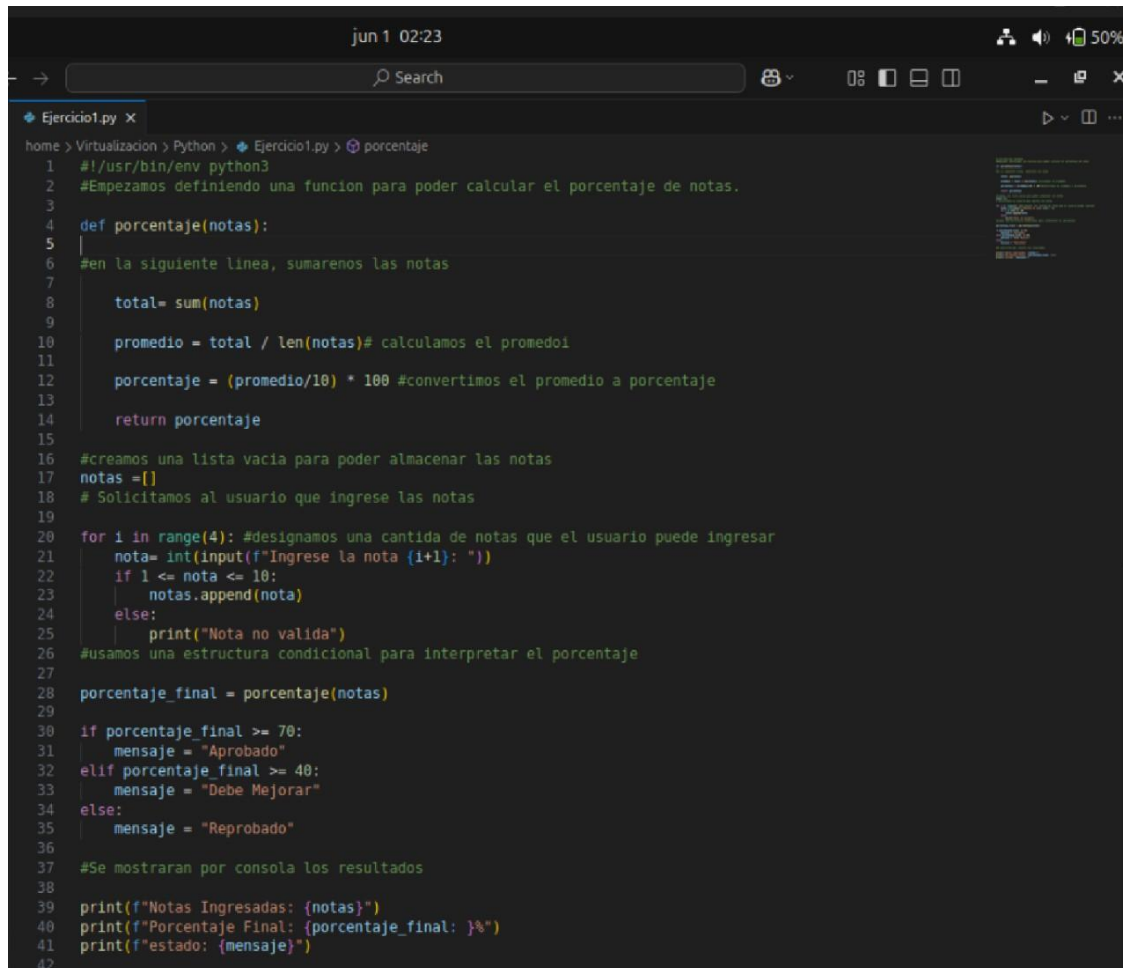
• Creamos una carpeta llamada “Python”

• Accedemos a la carpeta creada

• Creamos un archivo llamado “Ejercicio1.py”



- Desarrollamos un código en Python para poder calcular el promedio de las notas ingresadas.

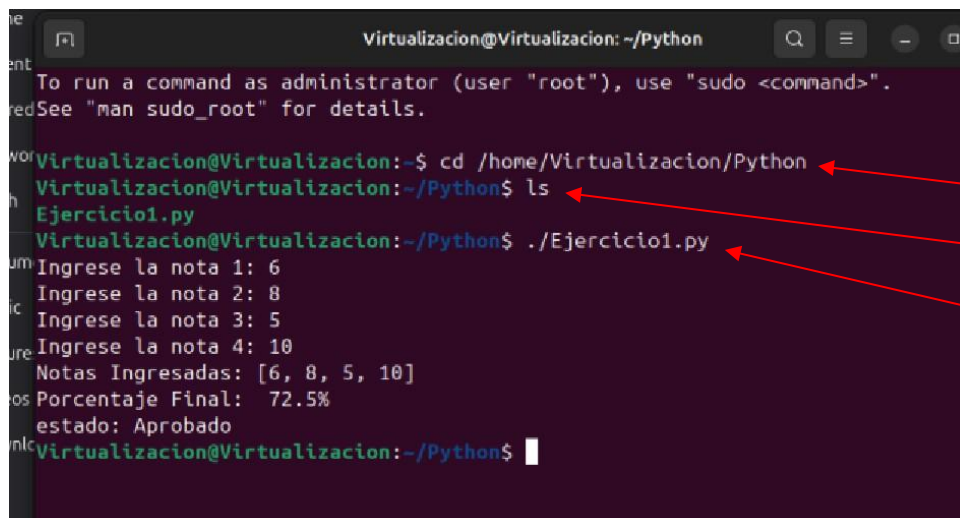


```

home > Virtualizacion > Python > Ejercicio1.py > porcentaje
1  #!/usr/bin/env python3
2  #Empezamos definiendo una funcion para poder calcular el porcentaje de notas.
3
4  def porcentaje(notas):
5  |
6  #en la siguiente linea, sumaremos las notas
7
8      total= sum(notas)
9
10     promedio = total / len(notas)# calculamos el promedio
11
12     porcentaje = (promedio/10) * 100 #convertimos el promedio a porcentaje
13
14     return porcentaje
15
16 #creamos una lista vacia para poder almacenar las notas
17 notas =[]
18 # Solicitamos al usuario que ingrese las notas
19
20 for i in range(4): #designamos una cantidad de notas que el usuario puede ingresar
21     nota= int(input(f"Ingrese la nota {i+1}: "))
22     if 1 <= nota <= 10:
23         notas.append(nota)
24     else:
25         print("Nota no valida")
26 #usamos una estructura condicional para interpretar el porcentaje
27
28 porcentaje_final = porcentaje(notas)
29
30 if porcentaje_final >= 70:
31     mensaje = "Aprobado"
32 elif porcentaje_final >= 40:
33     mensaje = "Debe Mejorar"
34 else:
35     mensaje = "Reprobado"
36
37 #Se mostraran por consola los resultados
38
39 print(f"Notas Ingresadas: {notas}")
40 print(f"Porcentaje Final: {porcentaje_final:}%")
41 print(f"estado: {mensaje}")
42

```

- Ahora en la terminal ejecutamos y probamos el Código.



```

Virtualizacion@Virtualizacion: ~/Python
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Virtualizacion@Virtualizacion:~/Python$ cd /home/Virtualizacion/Python
Virtualizacion@Virtualizacion:~/Python$ ls
Ejercicio1.py
Virtualizacion@Virtualizacion:~/Python$ ./Ejercicio1.py
Ingrese la nota 1: 6
Ingrese la nota 2: 8
Ingrese la nota 3: 5
Ingrese la nota 4: 10
Notas Ingresadas: [6, 8, 5, 10]
Porcentaje Final: 72.5%
estado: Aprobado
Virtualizacion@Virtualizacion:~/Python$

```

- Nos ubicamos en la carpeta creada (comando "cd")
- Listamos los archivos y buscamos si está el que creamos. (comando "ls")
- Ejecutamos el archivo "Ejercicio1.py" e interactuamos a través de la Terminal.

4. Conclusión

Dado todo lo visto, podemos afirmar que la virtualización de máquinas es una herramienta clave para la experimentación y la implementación de tecnologías. Sirve como un entorno de prueba donde es posible crear y modificar configuraciones de sistemas a voluntad, así como explorar distintos escenarios sin poner en riesgo el sistema principal del usuario. Esta oportunidad de equivocarnos nos enseña a ser más cuidadosos a la hora de planificar una rutina secuencial para cualquier desarrollo, y además nos permite comprender las distintas circunstancias que pueden presentarse al enfrentarnos a algo nuevo.

Personalmente, nos costó un poco llevar a cabo este trabajo debido a algunas preferencias que dejamos de lado, además de la organización con la que contamos para realizarlo. Sin embargo, logramos concretar nuestro objetivo: ejecutar el programa en Ubuntu. El esfuerzo valió la pena, y nos sentimos satisfechos con nuestro trabajo; nos comprometemos a seguir el ritmo en la materia y alcanzar más metas como esta.

Nos llevamos un gran conocimiento que podremos implementar en un futuro cercano. Las máquinas virtuales llegaron para quedarse y serán herramientas importantes para realizar simulaciones y pruebas reales en futuros proyectos.

Vale la pena el tiempo invertido, que nos hizo comprender que existen formas más sencillas para elaborar una máquina virtual funcional y práctica. Usaremos todo el contenido que no exploramos, ya sea por falta de tiempo o distracciones, para crear otra VM con, por ejemplo, una versión menos pesada que Ubuntu.

Desde ya le recalcamos su gran empeño para elaborar los contenidos con los cuales hemos realizado el trabajo. ¡Muchas gracias!

5. Bibliografía.

- Coulouris, G., Dollimore, J., & Kindberg, T. (2019). Distributed Systems: Concepts and Design. Pearson.
- Deitel, H. M., & Deitel, P. J. (2019). Operating Systems. Pearson.
- Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts. John Wiley & Sons.
- Stallings, W. (2020). Operating Systems: Internals and Design Principles. Pearson.
- Stallings, W. (2023). Operating Systems: Internals and Design Principles. Pearson.
- Tanenbaum, A. S., & Bos, H. (2021). Modern Operating Systems. Pearson.

6. Anexo

- *Video de presentación del trabajo.*
- *Repositorio de GitHub donde se encuentra el trabajo completo.*