Virtualización con VirtualBox

Alumno:

Fernando Javier Elichiribehety - fernando.elichiribehety@tupad.utn.edu.ar

Materia: Arquitectura y Sistemas Operativos

Profesor: Osvaldo Falabella

Fecha de Entrega: 28/07/2025

Índice

Introducción

La virtualización es una tecnología que crea versiones "virtuales" (simuladas por software) de recursos que normalmente serían físicos. Lo más común es crear máquinas virtuales (VMs). Una VM es como una computadora completa que funciona dentro de tu computadora real: tiene su propio sistema operativo (Windows, Linux, etc.), sus programas y sus propios recursos (parte del procesador, la memoria, el disco duro de la máquina real).

Beneficios:

- **Aprovechar mejor el hardware:** En lugar de tener muchos servidores físicos que a menudo están subutilizados, la virtualización permite que un solo servidor ejecute varias VMs, usando mucho mejor su potencia.
- **Ahorrar dinero:** Menos servidores físicos significan menos gastos en compra, electricidad y mantenimiento.
- **Mayor flexibilidad:** Puedes crear, copiar o eliminar VMs muy fácilmente, lo que es ideal para probar software, desarrollar aplicaciones o mover servicios de un lugar a otro.
- **Aislamiento:** Si algo falla en una VM, no afecta a las otras VMs ni a la máquina física.

La virtualización nos permite ser más eficientes y flexibles con nuestros recursos informáticos, haciendo que una sola máquina física pueda funcionar como varias. Es la base de muchas de las tecnologías modernas que usamos hoy, como la computación en la nube.

Marco Teórico

La virtualización es una tecnología clave en la infraestructura de TI moderna, que permite la creación de entornos operativos simulados a partir de recursos de hardware. Su fundamento radica en la abstracción de los componentes físicos, facilitando una gestión más flexible y eficiente.

1. Hipervisores (Virtual Machine Monitor - VMM)

El **hipervisor** es el software o firmware esencial que permite la virtualización. Actúa como una capa entre el hardware físico y los sistemas operativos virtuales (llamados "Máquinas Virtuales" o "Máquinas Huésped"). Su función principal es gestionar los recursos de hardware y asignarlos dinámicamente a cada VM, asegurando que operen de forma aislada e independiente.

Existen dos tipos principales de hipervisores:

- **Hipervisor Tipo 1 (Bare-Metal):** Se instala directamente sobre el hardware físico del servidor, sin necesidad de un sistema operativo subyacente. Ofrece un alto rendimiento y seguridad, ya que interactúa directamente con el

hardware. Ejemplos incluyen VMware ESXi, Microsoft Hyper-V (en su rol de servidor) y Citrix XenServer. Son ideales para entornos de producción y centros de datos.

- **Hipervisor Tipo 2 (Hosted):** Se instala como una aplicación sobre un sistema operativo "host" tradicional (ej. Windows, Linux, macOS). Es más sencillo de configurar y usar para entornos de desarrollo, pruebas o uso personal. Sin embargo, su rendimiento es ligeramente inferior al Tipo 1, ya que las VMs acceden al hardware a través del sistema operativo anfitrión. Ejemplos populares son Oracle VirtualBox, VMware Workstation y Parallels Desktop.

2. Virtualización de Recursos

La virtualización no se limita solo a crear sistemas operativos completos; también se aplica a diferentes componentes del hardware para optimizar su uso:

- **Virtualización de Memoria:** Permite que las máquinas virtuales operen con la ilusión de tener memoria RAM dedicada, incluso si el hipervisor la gestiona y comparte dinámicamente desde la memoria física disponible. Técnicas como la paginación de segundo nivel y la desduplicación de memoria optimizan su uso.
- **Virtualización de Almacenamiento:** Abstrae el almacenamiento físico (discos duros, arrays RAID) de las VMs. Esto permite a las VMs ver discos virtuales, cuya capacidad puede ser asignada desde un pool de almacenamiento compartido. Facilita la movilidad de datos, la gestión centralizada y la recuperación ante desastres.
- **Virtualización de Red:** Crea redes virtuales, adaptadores de red virtuales y switches virtuales, permitiendo que las VMs se comuniquen entre sí y con redes externas de manera aislada y configurable. Permite la creación de topologías de red complejas en un único hardware físico.
- **Virtualización de Dispositivos (I/O):** Permite que múltiples VMs compartan dispositivos físicos de entrada/salida como tarjetas de red, GPUs o unidades de almacenamiento. Esto puede hacerse mediante emulación, paravirtualización o passthrough directo para maximizar el rendimiento.

3. Docker

Mientras que la virtualización tradicional (VMs) virtualiza el hardware subyacente, **Docker** representa una forma de **virtualización a nivel de sistema operativo**, conocida como **contenerización**.

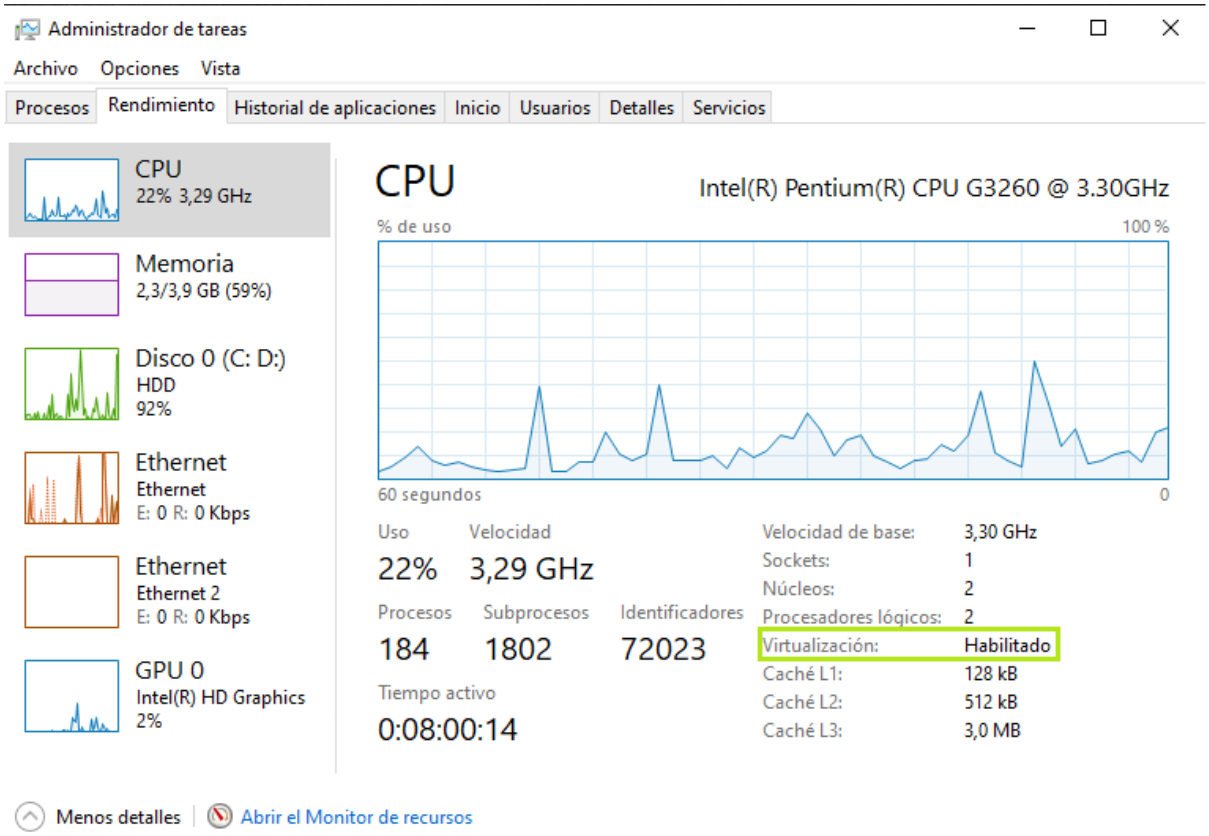
- **¿Qué es Docker?** Docker es una plataforma de código abierto que facilita la creación, el despliegue y la ejecución de aplicaciones utilizando **contenedores**. A diferencia de las VMs, que incluyen un sistema operativo completo, los contenedores Docker comparten el kernel del sistema operativo del host. Esto los hace mucho más ligeros, rápidos de iniciar y eficientes en el uso de recursos. Cada contenedor empaqueta una aplicación y todas sus dependencias (bibliotecas, configuraciones, etc.) en una unidad estandarizada que puede ejecutarse de manera consistente en cualquier entorno.
- **¿Para qué sirve Docker?**
    - **Desarrollo y Despliegue Consistentes:** Elimina el problema de "funciona en mi máquina, pero no en producción", ya que el entorno de la aplicación es idéntico en todas partes.
    - **Portabilidad:** Un contenedor se ejecuta de la misma manera en cualquier máquina que tenga Docker instalado.
    - **Aislamiento y Seguridad:** Las aplicaciones en diferentes contenedores están aisladas entre sí, evitando conflictos de dependencias.
    - **Escalabilidad:** Permite escalar aplicaciones rápidamente lanzando múltiples instancias del mismo contenedor.
    - **Eficiencia de Recursos:** Utiliza los recursos del sistema de manera mucho más eficiente que las VMs.
- **Docker Compose: Docker Compose** es una herramienta para definir y ejecutar aplicaciones Docker de múltiples contenedores. En lugar de ejecutar comandos docker run individuales para cada parte de la aplicación (por ejemplo, un servidor web, una base de datos, un cliente), Docker Compose permite definir todos los servicios de la aplicación en un único archivo YAML (docker-compose.yml). Esto simplifica enormemente el proceso de configurar, iniciar y detener aplicaciones complejas que consisten en varios

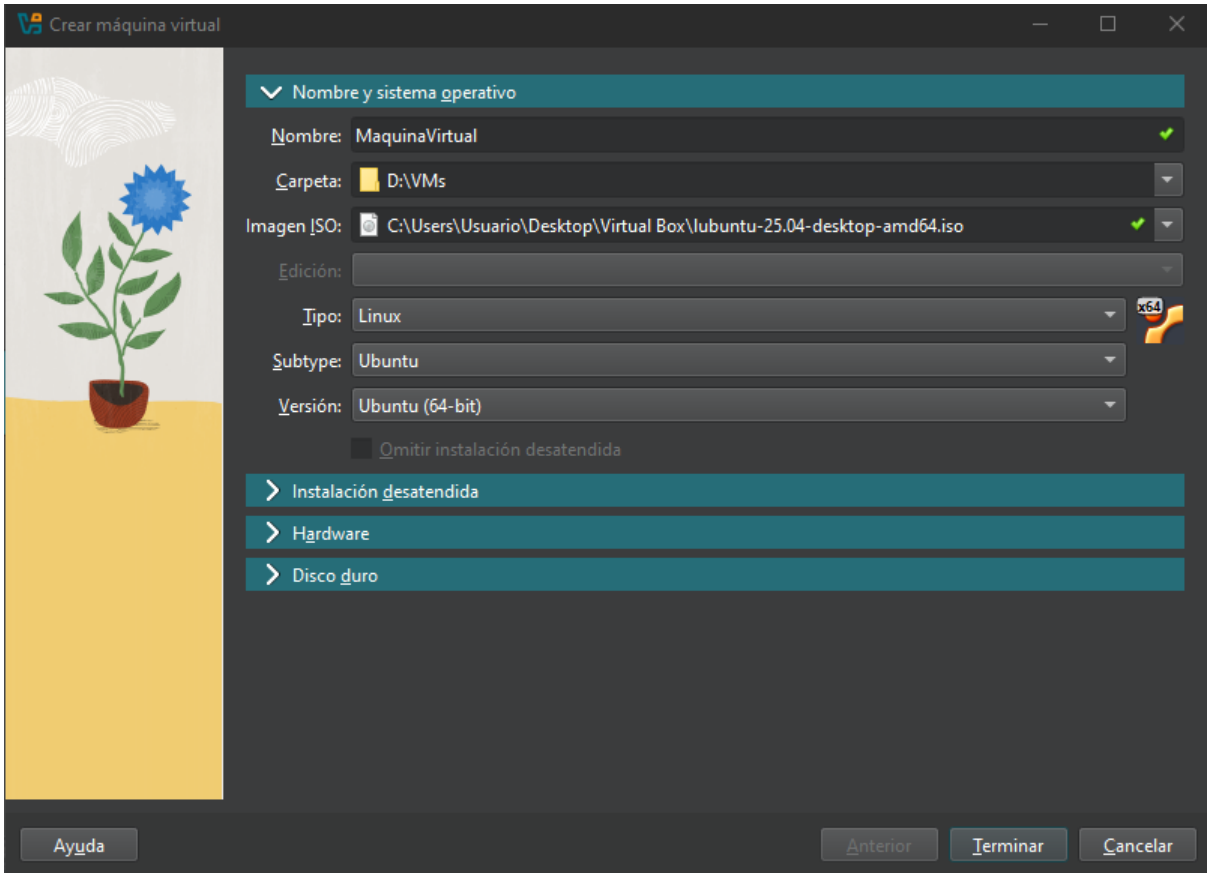
microservicios interconectados. Define cómo se construyen las imágenes, qué puertos se exponen, qué redes utilizan y qué dependencias existen entre los servicios.

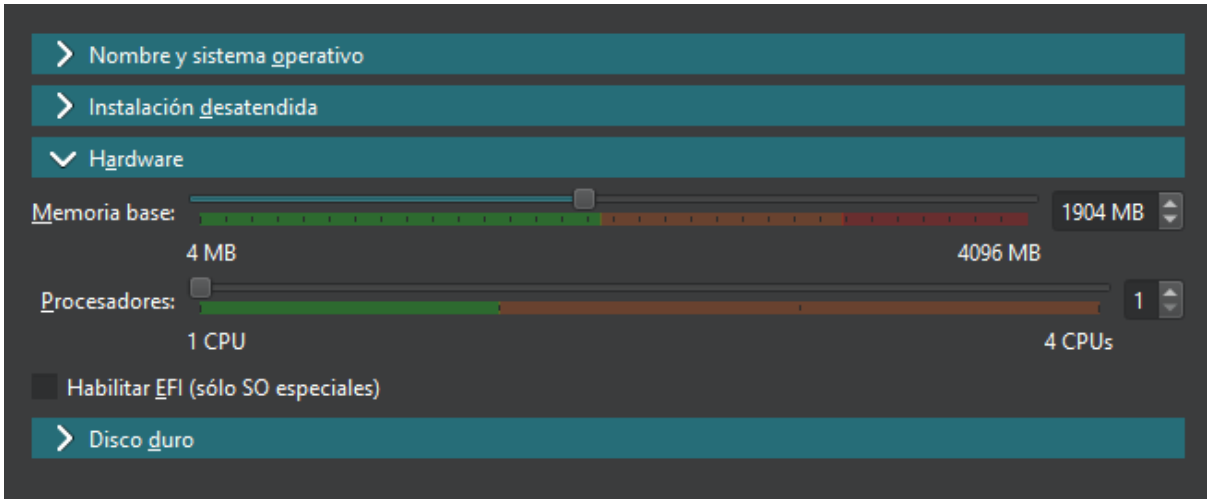
Caso Práctico

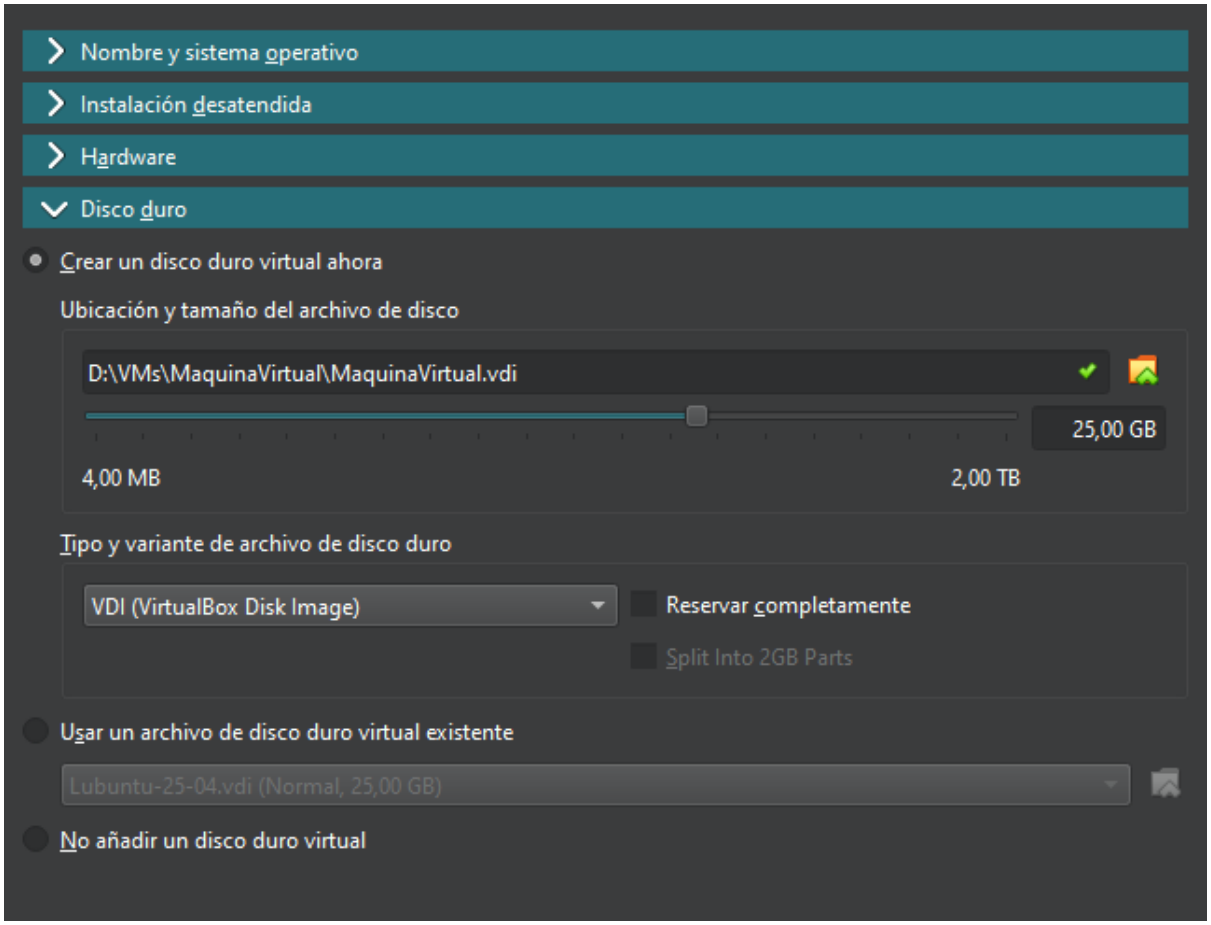
Instalación de una máquina virtual mediante VirtualBox:

- Verificar que la virtualización del CPU del host esté habilitada.
- Nombre de la máquina virtual.
- Carpeta de destino.
- Imagen ISO del sistema operativo elegido, en este caso Lubuntu 24.04.
- Asignación de recursos del hardware del host (memoria RAM, espacio en disco).

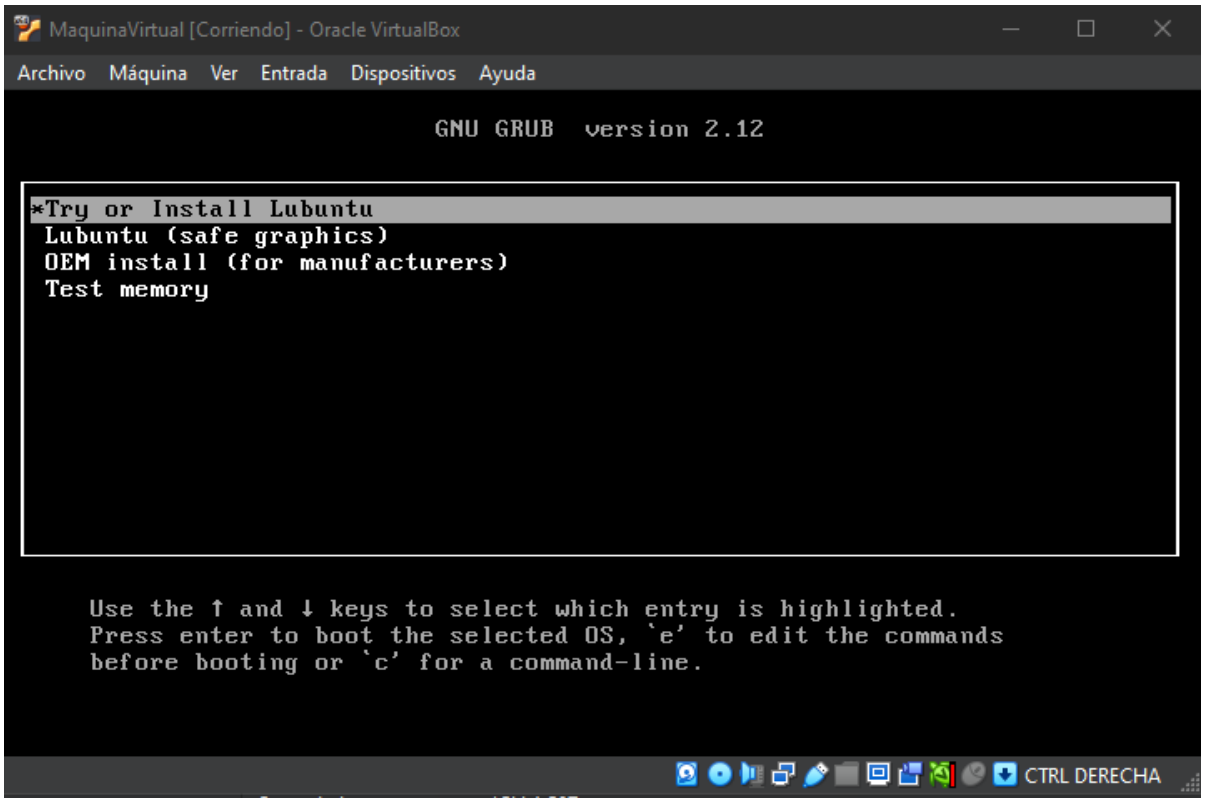


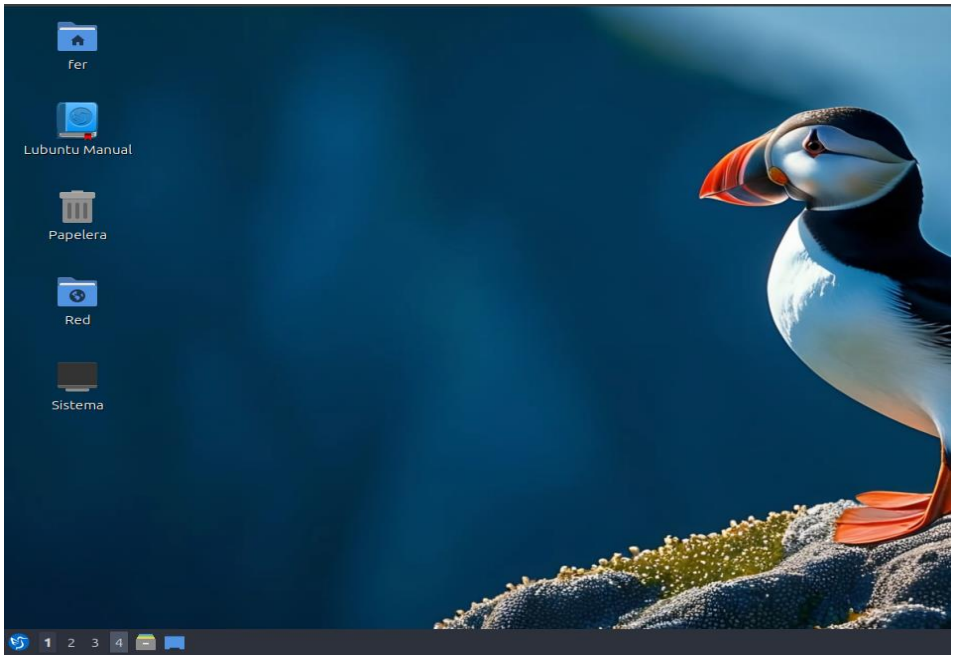

Crear máquina virtual
Nombre y sistema operativo
Nombre: MaquinaVirtual
Carpeta: D:\VMs
Imagen ISO: C:\Users\Usuario\Desktop\Virtual Box\lubuntu-25.04-desktop-amd64.iso
Edición:
Tipo: Linux
Subtype: Ubuntu
Versión: Ubuntu (64-bit)
Omitir instalación desatendida
Instalación desatendida
Hardware
Disco duro
Ayuda
Anterior
Terminar
Cancelar


Nombre y sistema operativo
Instalación desatendida
Hardware
Memoria base:
1904 MB
4 MB
4096 MB
Procesadores:
1
1 CPU
4 CPUs
Habilitar EFI (sólo SO especiales)
Disco duro


Nombre y sistema operativo
Instalación desatendida
Hardware
Disco duro
Crear un disco duro virtual ahora
Ubicación y tamaño del archivo de disco
D:\VMs\MaquinaVirtual\MaquinaVirtual.vdi
25,00 GB
4,00 MB
2,00 TB
Tipo y variante de archivo de disco duro
VDI (VirtualBox Disk Image)
Reservar completamente
Split Into 2GB Parts
Usar un archivo de disco duro virtual existente
Lubuntu-25-04.vdi (Normal, 25,00 GB)
No añadir un disco duro virtual

Instalación del Sistema Operativo:





Verificación de versión de Python e instalación del gestor de paquetes de Python, que permite descargar e instalar librerías, módulos y frameworks:

```
fer@fer-virtualbox:~$ python3 --version
Python 3.13.3
fer@fer-virtualbox:~$ sudo apt install python3-pip -y
Upgrading:
  libpython3.13  libpython3.13-minimal  libpython3.13-stdlib  python3.13  python3.13-gc

Installing:
  python3-pip

Installing dependencies:
  binutils                  g++-x86-64-linux-gnu       libasan8       libhwasan0
  binutils-common           gcc                        libbinutils    libitm1
  binutils-x86-64-linux-gnu gcc-14                     libcc1-0       libjs-jquery
  build-essential           gcc-14-x86-64-linux-gnu    libctf-nobfd0  libjs-sphinxdoc
  dpkg-dev                  gcc-x86-64-linux-gnu       libctf0        libjs-underscore
  fakeroot                  javascript-common          libexpat1-dev  liblsan0
  g++                       libalgorithm-diff-perl     libfakeroot    libpython3-dev
  g++-14                    libalgorithm-diff-xs-perl  libgcc-14-dev  libpython3.13-de
  g++-14-x86-64-linux-gnu   libalgorithm-merge-perl    libgprofng0    libquadmath0

Paquetes sugeridos:
  binutils-doc     g++-multilib     autoconf  bison            gcc-14-locales        | h
  gprofng-gui      g++-14-multilib  automake  gdb              gdb-x86-64-linux-gnu  lib
  binutils-gold    gcc-14-doc       libtool   gcc-doc          apache2               mak
  debian-keyring   gcc-multilib     flex      gcc-14-multilib  | lighttpd           pyt

Summary:
  Upgrading: 6, Installing: 46, Removing: 0, Not Upgrading: 181
  Download size: 75,1 MB
  Space needed: 244 MB / 19,5 GB available
```

Ejemplo de instalación de la librería requests para peticiones HTTP:

```
fer@fer-virtualbox:~$ sudo apt install python3-requests
[sudo] password for fer:
Upgrading:
  python3-requests

Summary:
  Upgrading: 1, Installing: 0, Removing: 0, Not Upgrading: 180
  Download size: 53,0 kB
  Space needed: 0 B / 19,2 GB available

Des:1 http://archive.ubuntu.com/ubuntu plucky-updates/main amd64 python3-requests all 2
B]
Descargados 53,0 kB en 1s (49,4 kB/s)
(Leyendo la base de datos ... 257318 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../python3-requests_2.32.3+dfsg-4ubuntu1.1_all.deb ...
Desempaquetando python3-requests (2.32.3+dfsg-4ubuntu1.1) sobre (2.32.3+dfsg-4ubuntu1)
Configurando python3-requests (2.32.3+dfsg-4ubuntu1.1) ...
```

Ejemplo de instalación de navegador Firefox:

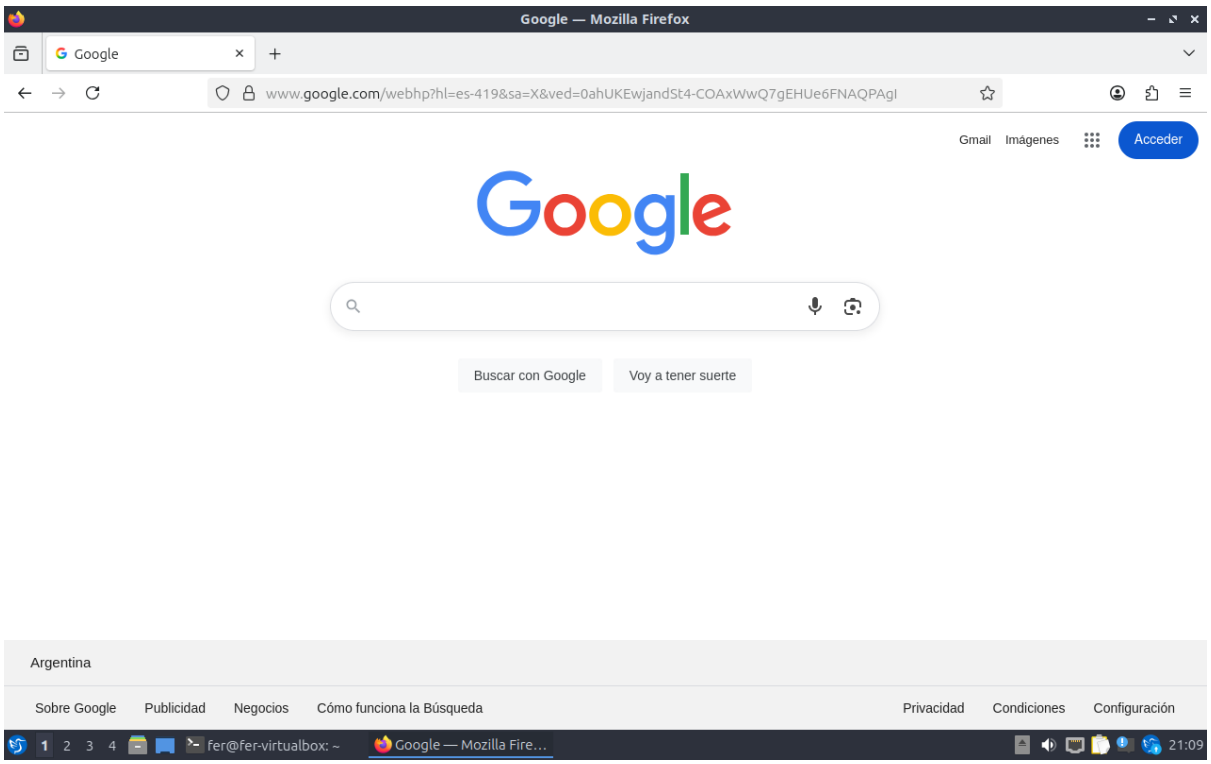
```
fer@fer-virtualbox:~$ sudo apt update
Obj:1 http://archive.ubuntu.com/ubuntu plucky InRelease
Obj:2 http://archive.ubuntu.com/ubuntu plucky-updates InRelease
Obj:3 http://security.ubuntu.com/ubuntu plucky-security InRelease
Obj:4 http://archive.ubuntu.com/ubuntu plucky-backports InRelease
Se pueden actualizar 180 paquetes. Ejecute «apt list --upgradable» para verlos.
fer@fer-virtualbox:~$ sudo apt install firefox -y
Installing:
  firefox

Installing dependencies:
  snapd  squashfs-tools

Paquetes sugeridos:
  zenity  | kdialog

Summary:
  Upgrading: 0, Installing: 3, Removing: 0, Not Upgrading: 180
  Download size: 32,2 MB
  Space needed: 122 MB / 19,2 GB available

Des:1 http://archive.ubuntu.com/ubuntu plucky/main amd64 squashfs-tools amd64 1:4
Des:2 http://archive.ubuntu.com/ubuntu plucky/main amd64 snapd amd64 2.67.1+25.04
Des:3 http://archive.ubuntu.com/ubuntu plucky/main amd64 firefox amd64 1:1snap1-0
Descargados 32,2 MB en 4s (7.207 kB/s)
Preconfigurando paquetes ...
Seleccionando el paquete squashfs-tools previamente no seleccionado.
```



Metodología Utilizada

- Instalación de VirtualBox en Windows.
- Descarga de imagen ISO oficial de Ubuntu.

- Instalación del Sistema Operativo.
- Verificación e Instalación de Python y gestor de paquetes.
- Ejemplo de instalación de programas.

Resultados Obtenidos

- **Entorno de Virtualización Operativo:** VirtualBox fue instalado correctamente en Windows, y la máquina virtual con Lubuntu 25.04 se configuró y ejecutó de forma estable, demostrando el sistema operativo invitado completamente funcional. La red en modo puente permitió el acceso desde el exterior.
- **Preparación de Entorno de Desarrollo:** Python 3 y su gestor de paquetes pip se verificaron e instalaron satisfactoriamente en el sistema invitado, creando una base sólida para el desarrollo de aplicaciones.
- **Demostración de Utilidad:** Se validó la capacidad de la máquina virtual para navegar por internet (ej. Firefox) y para gestionar software (ej. instalación de requests y firefox vía apt), destacando el aislamiento y la versatilidad del entorno virtualizado.

Conclusiones

Este trabajo práctico permitió una comprensión integral de la virtualización y sus beneficios, a través de la implementación y gestión de una máquina virtual.

Se demostró la capacidad de la virtualización para optimizar el hardware, permitir el aislamiento de sistemas operativos invitados y facilitar la experimentación con software sin afectar el sistema host. La instalación de Lubuntu como sistema operativo invitado, junto con la verificación y el uso de Python 3 y su gestor de paquetes, demostró la viabilidad de configurar un entorno de desarrollo completo y funcional dentro de un ambiente virtualizado. Además, la capacidad de instalar y ejecutar aplicaciones adicionales como Firefox y librerías Python (ej. requests) validó la flexibilidad y la versatilidad que ofrece una máquina virtual para diversas tareas.

En definitiva, la virtualización se consolida como una herramienta fundamental en la arquitectura de sistemas operativos modernos, ofreciendo flexibilidad, eficiencia y robustez para diversos escenarios de desarrollo y pruebas.

Anexos

1. Carpeta digital: Repositorio Git Hub

2. Video explicativo: Video Youtube