

Práctica de Laboratorio 1

Instalación y Configuración de Máquinas Virtuales

Estudiantes:

Wílmer E. León
Código: 1520010896

Jesús Orlando Orjuela
Código: 100384722

Hugo Alejandro Mejía
Código: 100312289

Fabián Andrés Cabana
Código: 1620010455

Estudiante cinco
Código: XXXXXXXX

Docente:

José León León

Institución Universitaria Politécnico Grancolombiano
Facultad de Ingeniería, Diseño e Innovación
Sistemas Operacionales - Grupo B04 | Grupo de trabajo 11
Bogotá D.C., Colombia
1 de noviembre de 2025

Índice

1. Introducción	2
2. Marco teórico	2
2.1. NGiNX como balanceador de carga	3
3. Contenido	4
3.1. Preparación de los recursos	4
3.2. Creación de las máquinas virtuales	4
3.3. Configuración de red	5
3.4. Evidencia de configuración de red	5
3.5. Evidencia de conectividad entre máquinas	7
3.6. Diagramas de red	9
4. Implementación con Docker	9
5. Conclusiones	10
6. Referencias	10
Referencias	10

1. Introducción

En esta primera entrega se documenta la instalación de tres máquinas virtuales utilizando **Oracle VirtualBox**, su configuración de red en modo **adaptador puente** y la verificación de conectividad entre ellas. El objetivo es sentar las bases para la posterior configuración de **Nginx** como balanceador de carga.

Como señalan autores como Stallings (2005), la virtualización es un pilar fundamental en la administración moderna de sistemas. Asimismo, Silberschatz, Galvin, y Gagne (2008) destacan que la concurrencia y la gestión de procesos son elementos centrales en el diseño de sistemas operativos.

En el Marco teórico, se define el respaldo conceptual y antecedentes que respaldan el montaje de las actividades para esta entrega, junto con las bibliografías que lo apoyan.

En el Contenido, se detallan paso a paso la preparación de los recursos, creación de máquinas virtuales, configuración de red y todas las evidencias con capturas de pantalla. Finalmente, se plantea una alternativa de creación de máquinas virtuales por Docker, solo como un plus para facilitar la articulación de metodologías en un grupo de trabajo para esta y posteriores entregas

2. Marco teórico

La virtualización se ha consolidado como una tecnología fundamental en la administración de sistemas modernos. Según Wolf, Ruiz, Bergero, y Meza (2010), la virtualización permite optimizar recursos y aislar entornos de ejecución, lo que facilita la administración de sistemas. Esta capacidad de crear múltiples sistemas operativos independientes sobre un único hardware físico ha revolucionado la forma en que se gestionan los recursos computacionales en entornos empresariales y académicos.

Stallings (2005) explica que la virtualización no es un concepto nuevo, pero su implementación moderna mediante hipervisores ha permitido alcanzar niveles de eficiencia sin precedentes. En el contexto de este laboratorio, el montaje de tres sistemas operativos desde distintas máquinas se efectúa para hacer pruebas de conectividad y simular escenarios de red realistas sin necesidad de hardware adicional.

En cuanto a las configuraciones de red en entornos virtualizados, existen diferencias clave entre los modos NAT (Network Address Translation) y adaptador puente en VirtualBox. Oracle (2024) documentan que estas diferencias determinan el alcance de la conectividad de las máquinas virtuales. Mientras que el modo NAT proporciona aislamiento y permite que las VMs accedan a redes externas a través del host, el modo adaptador puente integra las VMs directamente en la red física, asignándoles direcciones IP propias y permitiendo comunicación directa con otros

dispositivos de la red local.

El direccionamiento IP es esencial para garantizar la comunicación entre dispositivos en una red local. Como señala Canonical (2024) en la documentación oficial de Ubuntu Server, la configuración correcta de interfaces de red mediante herramientas como Netplan es fundamental para establecer conectividad estable y predecible. La asignación de direcciones IP estáticas, en contraposición al uso de DHCP, proporciona mayor control y facilita la configuración de servicios que requieren direcciones fijas, como será el caso del balanceador de carga Nginx en fases posteriores.

2.1. NGiNX como balanceador de carga

Nginx (pronunciado /'en-gin-ex) es un servidor web de alto rendimiento y proxy inverso que se ha convertido en una herramienta estándar para la distribución de carga en arquitecturas de servidores modernos. Su arquitectura basada en eventos le permite manejar miles de conexiones simultáneas con un consumo mínimo de recursos, lo que lo hace ideal para entornos virtualizados.

En el contexto de este laboratorio, la configuración de red en modo adaptador puente entre las tres máquinas virtuales sienta las bases para implementar Nginx como balanceador de carga en entregas futuras. El balanceo de carga permite distribuir el tráfico de red entre múltiples servidores, mejorando la disponibilidad, escalabilidad y rendimiento de las aplicaciones. Nginx puede configurarse para distribuir peticiones HTTP/HTTPS entre los servidores backend utilizando diferentes algoritmos (round-robin, least connections, ip-hash), permitiendo que si uno de los servidores falla, el tráfico se redirija automáticamente a los servidores restantes.

La instalación de Nginx en Ubuntu Server es sencilla mediante el gestor de paquetes APT, y su configuración se realiza principalmente a través de archivos en el directorio `/etc/nginx/`. Para el balanceo de carga, se define un grupo de servidores backend (upstream) con las direcciones IP de las máquinas virtuales configuradas en este laboratorio, estableciendo así la comunicación entre los tres sistemas mediante las pruebas de conectividad previamente validadas con el comando `ping`.

Silberschatz y cols. (2008) destacan que la concurrencia y la gestión de procesos son elementos centrales en el diseño de sistemas operativos, aspectos que se vuelven particularmente relevantes en entornos virtualizados donde múltiples sistemas operativos comparten recursos físicos. La correcta asignación de recursos (CPU, RAM, almacenamiento) entre las máquinas virtuales es crucial para garantizar el rendimiento óptimo de cada sistema.

Finalmente, Arena (2002) y Arena (2005) destacan la relevancia de Linux como sistema operativo robusto para servidores y entornos académicos, subrayando su estabilidad, seguridad y flexibilidad para configuraciones de red avanzadas. Ubuntu

Server, la distribución utilizada en este laboratorio, hereda estas características y ofrece soporte a largo plazo (LTS) que garantiza actualizaciones de seguridad y estabilidad durante varios años.

3. Contenido

3.1. Preparación de los recursos

Se utilizaron imágenes ISO de **Ubuntu Server**, en particular la versión **22.04.5 LTS** para la máquina **UbunSO2**. El hipervisor empleado fue **Oracle VirtualBox** (Oracle, 2024).

3.2. Creación de las máquinas virtuales

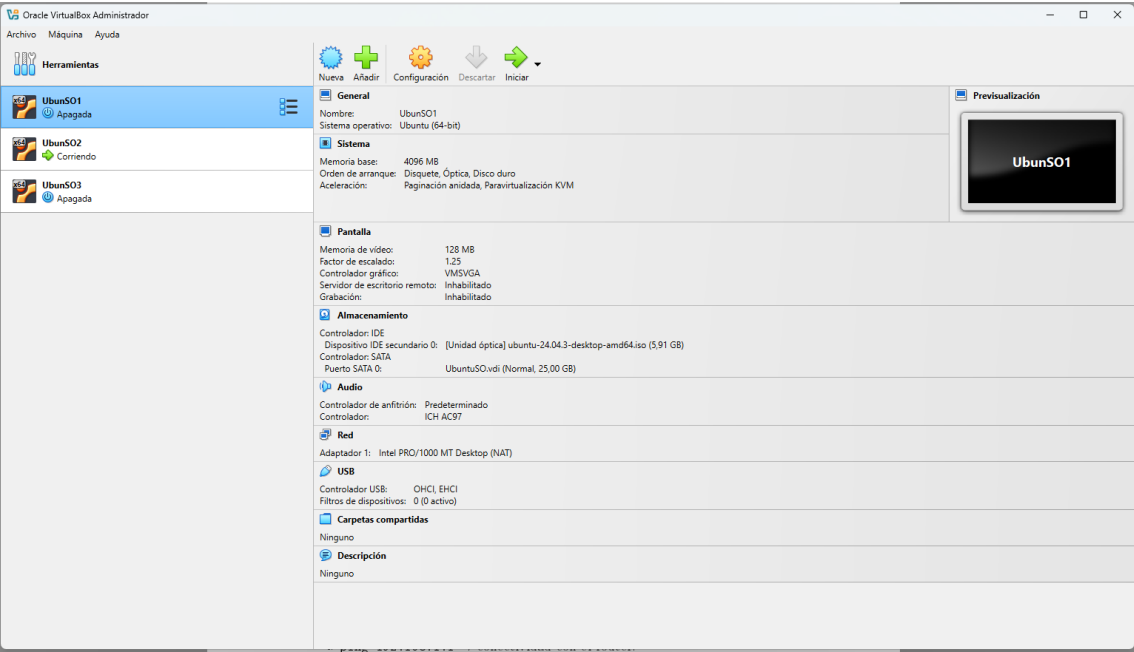


Figura 3.1. Evidencia de las tres máquinas virtuales creadas en VirtualBox.

3.3. Configuración de red

Tabla 3.1. *Resumen de configuración de las máquinas virtuales.*

Máquina	Nombre	Sistema Operativo	RAM (MB)	Dirección IP
UbunSO1	Ubuntu Server 22.04	Ubuntu Server 22.04 LTS	2048	192.168.2.9
UbunSO2	Ubuntu Server 22.04.5	Ubuntu Server 22.04.5 LTS	2048	192.168.2.8
UbunSO3	Ubuntu Server 22.04	Ubuntu Server 22.04 LTS	2048	192.168.2.7

3.4. Evidencia de configuración de red

Para verificar la configuración de red de cada máquina virtual, se utilizó el comando `ip a` (abreviatura de `ip address`) en cada una de las tres máquinas. Este comando muestra información detallada sobre las interfaces de red, incluyendo las direcciones IP asignadas, máscaras de subred y estado de las interfaces (es habitual que hayan fallas en la ejecución de los comandos, debido a falta de paquetes o a problemas con la instalación general del SO).

Pasos realizados:

1. Acceder a cada máquina virtual a través de la consola de VirtualBox.
2. Ejecutar el comando: `ip a` o `ip address show`
3. Verificar que la interfaz de red (típicamente `enp0s3`) tenga asignada la dirección IP configurada.
4. Confirmar que el estado de la interfaz sea `UP` y que esté en modo `BROADCAST,MULTICAST`.

A continuación se muestran las capturas de pantalla con la salida del comando en cada máquina virtual:

```
ubuntu@ubuntu-VirtualBox: ~  
$ ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host noprefixroute  
        valid_lft forever preferred_lft forever  
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000  
    link/ether 08:00:27:62:ba:ed brd ff:ff:ff:ff:ff:ff  
    inet 192.168.2.9/24 brd 192.168.2.255 scope global noprefixroute enp0s3  
        valid_lft forever preferred_lft forever  
    inet6 fe80::a00:27ff:fe62:baed/64 scope link dadfailed tentative  
        valid_lft forever preferred_lft forever  
ubuntu@ubuntu-VirtualBox: $
```

(a) *UbunSO1* – IP 192.168.2.9

```
ubuntu@ubuntu-VirtualBox: ~  
$ ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host noprefixroute  
        valid_lft forever preferred_lft forever  
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000  
    link/ether 08:00:27:62:ba:ed brd ff:ff:ff:ff:ff:ff  
    inet 192.168.2.8/24 brd 192.168.2.255 scope global noprefixroute enp0s3  
        valid_lft forever preferred_lft forever  
    inet6 fe80::a00:27ff:fe62:baed/64 scope link  
        valid_lft forever preferred_lft forever  
ubuntu@ubuntu-VirtualBox: $
```

(b) *UbunSO2* – IP 192.168.2.8

```
ubuntu@ubuntu-VirtualBox: ~  
$ ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host noprefixroute  
        valid_lft forever preferred_lft forever  
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000  
    link/ether 08:00:27:62:ba:ed brd ff:ff:ff:ff:ff:ff  
    inet 192.168.2.7/24 brd 192.168.2.255 scope global noprefixroute enp0s3  
        valid_lft forever preferred_lft forever  
    inet6 fe80::a00:27ff:fe62:baed/64 scope link  
        valid_lft forever preferred_lft forever  
ubuntu@ubuntu-VirtualBox: $
```

(c) *UbunSO3* – IP 192.168.2.7

Figura 3.2. Salida del comando *ip a* en las tres máquinas virtuales.

3.5. Evidencia de conectividad entre máquinas

Una vez verificada la configuración de red, se procedió a comprobar la conectividad entre las máquinas virtuales utilizando el comando `ping`. Este comando envía paquetes ICMP (Internet Control Message Protocol) para verificar si existe comunicación entre dos dispositivos en la red.

Procedimiento para las pruebas de conectividad:

1. Desde UbunSO1 hacia UbunSO2:

- Acceder a la máquina UbunSO1
- Ejecutar: `ping 192.168.2.8 -c 4`
- El parámetro `-c 4` indica que se enviarán 4 paquetes
- Verificar que se reciban respuestas (reply) y no haya pérdida de paquetes

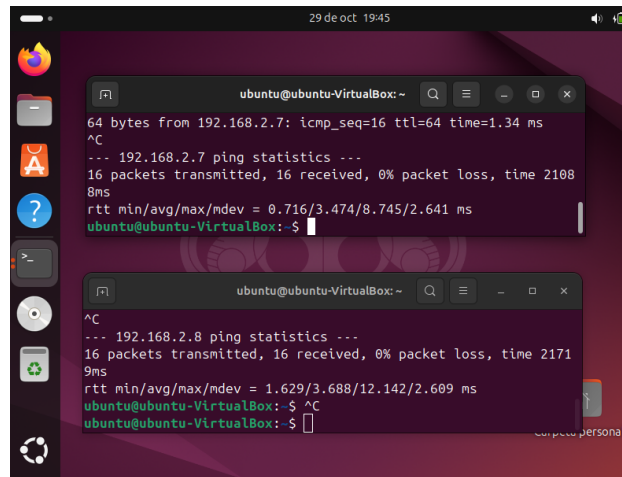
2. Desde UbunSO2 hacia UbunSO3:

- Acceder a la máquina UbunSO2
- Ejecutar: `ping 192.168.2.7 -c 4`
- Confirmar respuestas exitosas y tiempo de respuesta (TTL)

3. Desde UbunSO3 hacia UbunSO1:

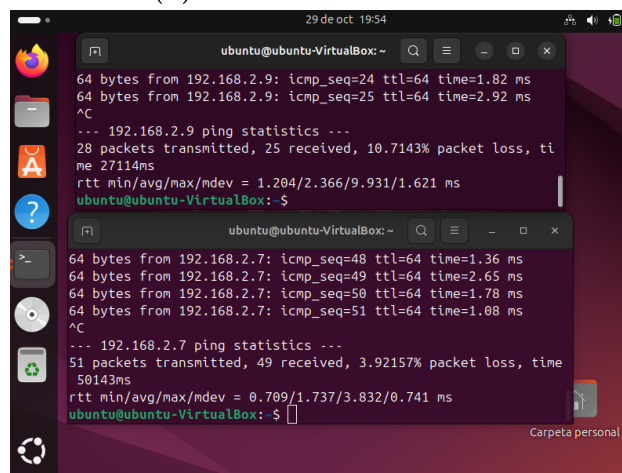
- Acceder a la máquina UbunSO3
- Ejecutar: `ping 192.168.2.9 -c 4`
- Validar conectividad bidireccional entre todas las máquinas

Las siguientes capturas muestran los resultados de las pruebas de conectividad, donde se observa el tiempo de respuesta (time), el TTL (Time To Live) y la estadística final con 0 % de pérdida de paquetes:



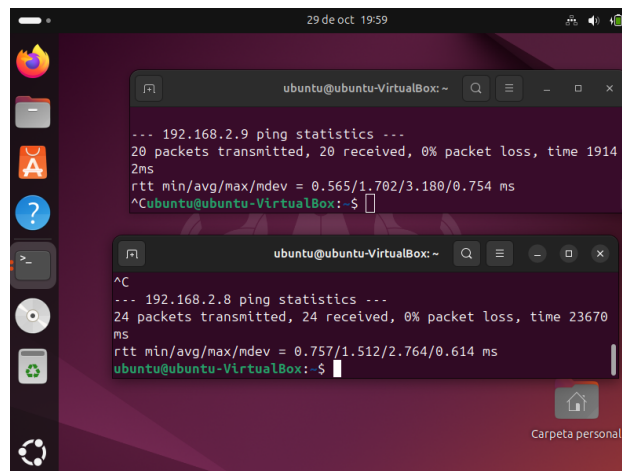
```
ubuntu@ubuntu-VirtualBox: ~  
64 bytes from 192.168.2.7: icmp_seq=16 ttl=64 time=1.34 ms  
^C  
--- 192.168.2.7 ping statistics ---  
16 packets transmitted, 16 received, 0% packet loss, time 2108  
8ms  
rtt min/avg/max/mdev = 0.716/3.474/8.745/2.641 ms  
ubuntu@ubuntu-VirtualBox:~$
```

(a) *UbunSO1* \rightarrow *UbunSO2*



```
ubuntu@ubuntu-VirtualBox: ~  
64 bytes from 192.168.2.9: icmp_seq=24 ttl=64 time=1.82 ms  
64 bytes from 192.168.2.9: icmp_seq=25 ttl=64 time=2.92 ms  
^C  
--- 192.168.2.9 ping statistics ---  
28 packets transmitted, 25 received, 10.7143% packet loss, ti  
me 27114ms  
rtt min/avg/max/mdev = 1.204/2.366/9.931/1.621 ms  
ubuntu@ubuntu-VirtualBox:~$  
64 bytes from 192.168.2.7: icmp_seq=48 ttl=64 time=1.36 ms  
64 bytes from 192.168.2.7: icmp_seq=49 ttl=64 time=2.65 ms  
64 bytes from 192.168.2.7: icmp_seq=50 ttl=64 time=1.78 ms  
64 bytes from 192.168.2.7: icmp_seq=51 ttl=64 time=1.08 ms  
^C  
--- 192.168.2.7 ping statistics ---  
51 packets transmitted, 49 received, 3.92157% packet loss, time  
50143ms  
rtt min/avg/max/mdev = 0.709/1.737/3.832/0.741 ms  
ubuntu@ubuntu-VirtualBox:~$
```

(b) *UbunSO2* \rightarrow *UbunSO3*



```
ubuntu@ubuntu-VirtualBox: ~  
--- 192.168.2.9 ping statistics ---  
20 packets transmitted, 20 received, 0% packet loss, time 1914  
2ms  
rtt min/avg/max/mdev = 0.565/1.702/3.180/0.754 ms  
^C  
ubuntu@ubuntu-VirtualBox:~$  
^C  
--- 192.168.2.8 ping statistics ---  
24 packets transmitted, 24 received, 0% packet loss, time 23670  
ms  
rtt min/avg/max/mdev = 0.757/1.512/2.764/0.614 ms  
ubuntu@ubuntu-VirtualBox:~$
```

(c) *UbunSO3* \rightarrow *UbunSO1*

Figura 3.3. Pruebas de conectividad entre las máquinas virtuales mediante *ping*.

3.6. Diagramas de red

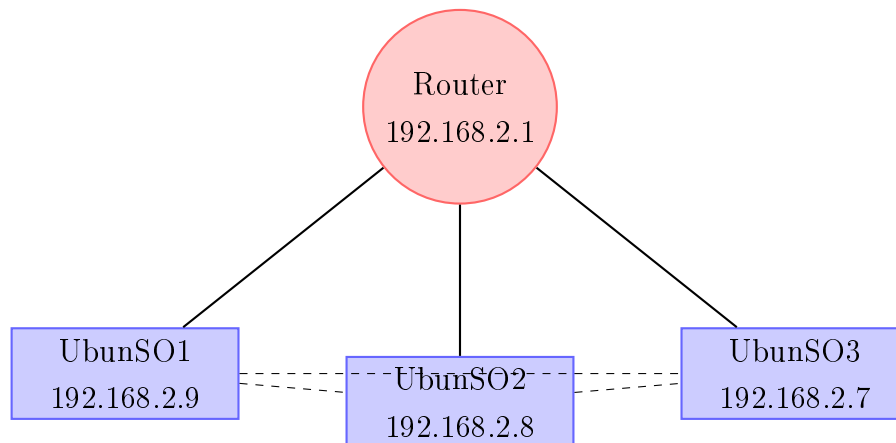


Figura 3.4. Comunicación entre las máquinas y el router en modo Adaptador Puen-te.

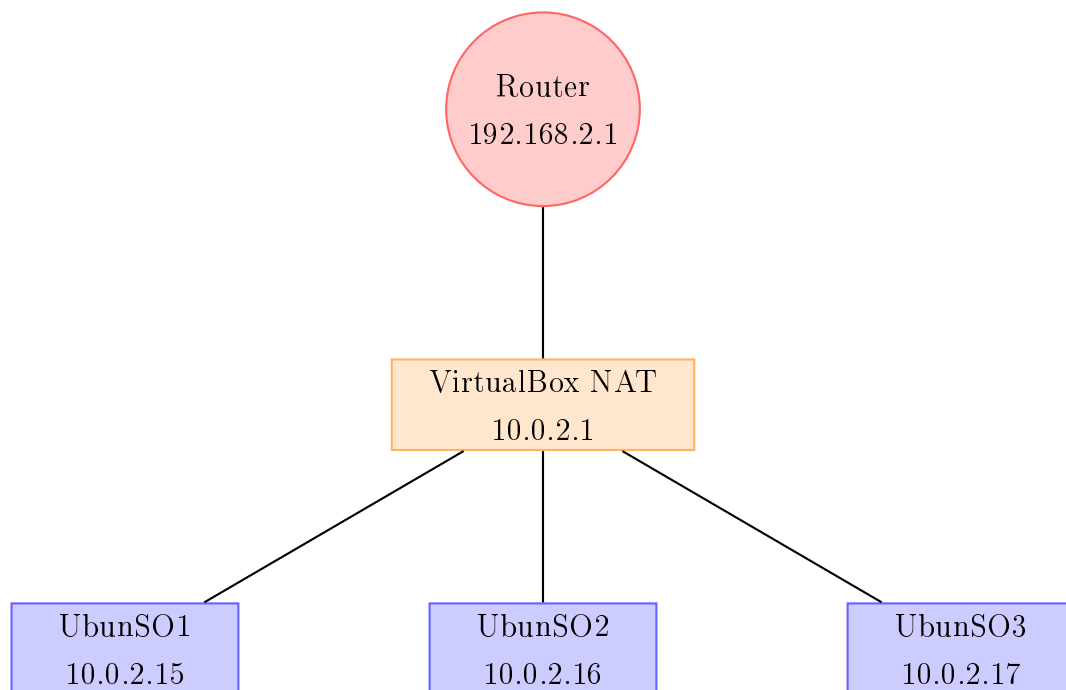


Figura 3.5. Esquema de red en modo NAT (referencia).

4. Implementación con Docker

Generamos además una implementación aprovechando la potencia de Docker para generar virtualización a partir de sus contenedores y teniendo en cuenta la importancia que tiene esta herramienta en el mundo profesional. Se generan contenedores y red via bash y podemos verlos ejecutando Ubuntu 22.04:

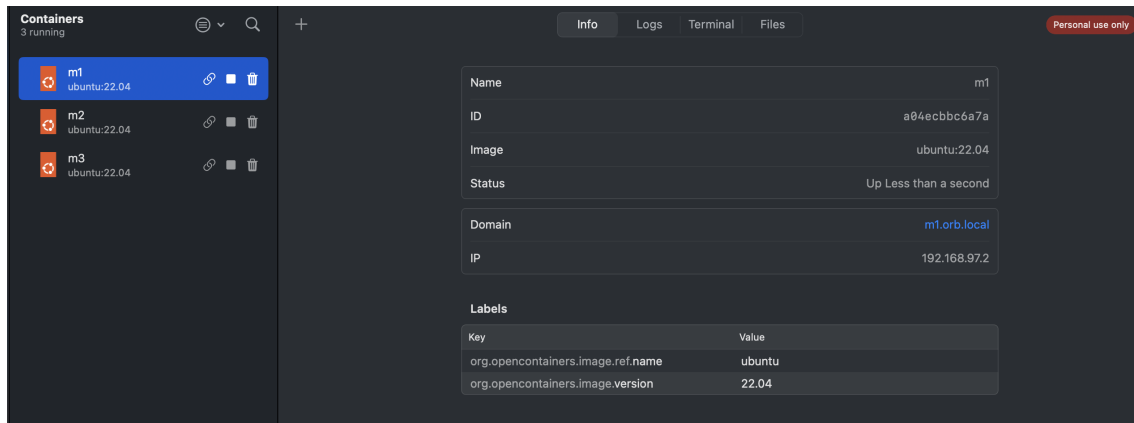


Figura 4.1. *Captura del software OrbStack*

5. Conclusiones

La configuración en modo puente permitió que cada VM tuviera una IP propia en la red local, facilitando la comunicación entre ellas y con el router. Las pruebas de `ping` confirmaron la conectividad entre las máquinas, sin pérdida de paquetes significativa. La asignación de IPs fijas mediante Netplan garantiza estabilidad para futuras pruebas y la siguiente fase con Nginx como balanceador de carga.

De acuerdo con Canonical (2024), la documentación oficial de Ubuntu recomienda este tipo de configuraciones para entornos de laboratorio. Asimismo, Oracle (2024) enfatiza la importancia de comprender las diferencias entre NAT y adaptador puente en VirtualBox para escenarios de red. Por su parte, Arena (2002) y Arena (2005) destacan la relevancia de Linux como sistema operativo robusto para servidores y entornos académicos, mientras que Wolf y cols. (2010) subraya su papel en la enseñanza universitaria.

6. Referencias

Referencias

- Arena, H. F. (2002). *La biblia de linux*. Buenos Aires: MP Ediciones.
- Arena, H. F. (2005). *La biblia de linux: Manual de uso, instalación y configuración*. Buenos Aires: RedUsers.
- Canonical. (2024). Ubuntu server documentation [Manual de software informático]. Descargado de <https://ubuntu.com/server/docs>
- Oracle. (2024). Virtualbox user manual [Manual de software informático]. Descargado de <https://www.virtualbox.org/manual/>

- Silberschatz, A., Galvin, P. B., y Gagne, G. (2008). *Fundamentos de sistemas operativos* (7a ed. ed.). México: McGraw-Hill.
- Stallings, W. (2005). *Sistemas operativos: Aspectos internos y principios de diseño* (5a ed. ed.). Madrid: Pearson Prentice Hall.
- Wolf, G., Ruiz, E., Bergero, F., y Meza, E. (2010). *Fundamentos de sistemas operativos*. México: Universidad Nacional Autónoma de México.