UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

FACULTAD DE INGENIERÍA ELECTRÓNICA

PROYECTO SEGUNDO CORTE

Desarrollo de un Entorno Híbrido de Pruebas para Análisis de Sistemas Operativos y Contenedores

Presentado a: Ing. Diego Alejandro Barragan Vargas
Juan Diego Báez Guerrero, Cód.: 2336781.

Resumen*— *Este documento presenta el desarrollo de un entorno híbrido de pruebas para sistemas operativos y arquitecturas de contenedores. Se analiza la instalación virtualizada de Rocky Linux, Manjaro Linux y Arch Linux mediante QEMU, así como contenedores Docker con imágenes de Garuda, Alpine y Debian. Un contenedor central con Fedora proporciona monitoreo de red y análisis de rendimiento del sistema mediante Grafana, Prometheus y Zabbix. El estudio evalúa la gestión de recursos, el comportamiento de la red y el monitoreo de servicios en diferentes arquitecturas de sistemas operativos.

Abstract*— *This paper presents the development of a hybrid testing environment for operating systems and container-based infrastructures. The analysis includes virtualized installations of Rocky Linux, Manjaro Linux, and Arch Linux using QEMU, as well as Docker containers with Garuda, Alpine, and Debian images. A central container system running Fedora provides network monitoring and system performance analysis using Grafana, Prometheus, and Zabbix. The study evaluates resource management, network behavior, and service monitoring in different operating system architectures.*

I. Introducción

En el ámbito de la administración de infraestructuras virtualizadas, los sistemas operativos y los contenedores han desempeñado un papel fundamental en la evolución de entornos computacionales. La virtualización permite el despliegue de múltiples sistemas sobre un único hardware físico, lo que optimiza recursos y facilita la gestión de entornos distribuidos [8]. En este estudio, se desarrollará un entorno híbrido de pruebas que integra máquinas virtuales y contenedores para evaluar el rendimiento y el monitoreo de sistemas operativos, empleando herramientas avanzadas de análisis.

Para la infraestructura de máquinas virtuales, se han seleccionado Rocky Linux, Manjaro Linux y Arch Linux, ejecutados mediante QEMU. Cada distribución tiene características particulares en la gestión de procesos y servicios, lo que permite examinar diferentes enfoques de administración [9]. La virtualización con QEMU ofrece una emulación eficiente del hardware, asegurando una separación efectiva de los entornos de prueba sin comprometer la estabilidad del sistema anfitrión [1]. Rocky Linux, Manjaro y Arch Linux cuentan con una comunidad activa y constantes mejoras, lo que los convierte en opciones ideales para entornos virtualizados [16], [17], [18].

Por otro lado, los contenedores han revolucionado la forma en que se ejecutan aplicaciones, proporcionando un aislamiento eficiente sin la sobrecarga de una máquina virtual completa [10]. Docker ha permitido la creación y gestión sencilla de entornos de ejecución de software, reduciendo problemas de compatibilidad entre sistemas [2]. En este estudio, se analizarán tres contenedores con imágenes de Garuda, Alpine y Debian, explorando sus capacidades de gestión y rendimiento [7].

El monitoreo de estos sistemas es clave para comprender su eficiencia en la administración de recursos [4]. Grafana y Prometheus han demostrado ser herramientas versátiles para la supervisión en tiempo real de métricas de CPU, red y almacenamiento [11], mientras que Zabbix permite gestionar alertas y eventos en entornos complejos [12]. La implementación de estas herramientas en el contenedor central basado en Fedora permitirá visualizar datos y detectar anomalías en los sistemas analizados.

En los sistemas Linux seleccionados, el control de servicios y procesos se gestiona principalmente con ‘systemd‘ y ‘journalctl‘, proporcionando herramientas eficaces para el registro y administración del estado del sistema [3]. En Rocky Linux, Manjaro y Arch Linux, estos comandos permiten optimizar la respuesta del sistema ante fallos y mejorar la administración de recursos [13].

El análisis de redes también desempeña un papel crucial en el estudio, permitiendo evaluar el tráfico generado entre los nodos del entorno híbrido [6]. Aplicaciones como Wireshark, ‘iftop‘ y ‘nethogs‘ ayudarán a visualizar y comprender las interacciones de los sistemas [14]. Además, herramientas como ‘nmap‘ y ‘netstat‘ contribuirán a la identificación de servicios activos y puertos abiertos en las máquinas virtuales y contenedores [15].

El almacenamiento y la administración de archivos son otro aspecto clave del estudio, ya que permiten una gestión eficiente de los datos utilizados en los sistemas analizados [13]. La combinación de herramientas como ‘ncdu‘ y ‘baobab‘ facilitará la identificación de patrones de uso de disco, mientras que ‘rsync‘ optimizará la sincronización de archivos y la transferencia de datos entre nodos [1].

Finalmente, este informe documentará el procedimiento de instalación, configuración y análisis de cada componente, destacando el impacto de la virtualización y contenedorización en la gestión de infraestructura distribuida. La implementación de un contenedor central para el monitoreo

permitirá una comparación detallada entre los distintos sistemas, proporcionando un análisis integral de la eficiencia y estabilidad de cada tecnología empleada.

II. Marco Teórico

A. Virtualización y Máquinas Virtuales

La virtualización ha revolucionado la infraestructura de TI al permitir la ejecución de múltiples sistemas en un mismo hardware físico [8]. Herramientas como QEMU [9] ofrecen emulación de hardware y administración eficiente de máquinas virtuales. Además, sistemas operativos como Rocky Linux, Manjaro y Arch Linux son ideales para analizar gestión de recursos en entornos virtualizados [1]. Cada distribución tiene su propia filosofía y enfoque en la administración del sistema, lo que las hace atractivas para pruebas comparativas [16], [17], [18].

B. Contenedores y Docker

Los contenedores proporcionan una alternativa ligera a la virtualización, aislando aplicaciones sin necesidad de hipervisores [10]. Docker ha sido fundamental en este avance, facilitando despliegues rápidos y portabilidad [2]. Los contenedores con sistemas como Garuda, Alpine y Debian permiten gestionar software sin impacto significativo en el rendimiento del host [7].

C. Gestión de Recursos en Sistemas Operativos

Cada sistema operativo administra sus recursos de manera distinta, afectando el rendimiento en entornos virtualizados y contenedorizados [1]. Herramientas como ‘glances‘, ‘bpytop‘ y ‘htop‘ permiten analizar el consumo de CPU, memoria y procesos activos en tiempo real [11].

D. Monitoreo de Sistemas

El monitoreo es crucial para la gestión de rendimiento y seguridad en infraestructuras virtualizadas [4]. Herramientas como Grafana y Prometheus recopilan métricas de CPU, red y almacenamiento [11], mientras que Zabbix se especializa en supervisión de servicios y alertas en entornos distribuidos [12].

E. Gestión de Servicios en Linux

La administración de procesos y servicios en sistemas Linux se basa en ‘systemd‘ y ‘journalctl‘, herramientas clave para la supervisión de eventos del sistema [3]. En Rocky Linux, Manjaro y Arch Linux, estos comandos permiten identificar fallas y mejorar la eficiencia del sistema [13].

F. Análisis de Redes y Seguridad

El monitoreo de tráfico es fundamental en entornos distribuidos [6]. Aplicaciones como Wireshark, ‘iftop‘ y ‘nethogs‘ permiten analizar el flujo de datos en la red [14]. Además, comandos como ‘nmap‘ y ‘netstat‘ facilitan la identificación de puertos abiertos y servicios activos en las máquinas virtuales y contenedores [15].

G. Administración de Almacenamiento y Archivos

La gestión eficiente del almacenamiento es esencial en entornos virtualizados [13]. Herramientas como ‘ncdu‘ y ‘baobab‘ permiten visualizar el uso del disco, mientras que ‘rsync‘ optimiza la sincronización de archivos [1].

H. Automatización y Gestión de Infraestructura

La administración automatizada de infraestructuras ha cobrado importancia con la adopción de herramientas como Ansible y Terraform [7]. Estas permiten configurar servidores y contenedores de manera repetible y eficiente.

I. Integración de Monitoreo con Contenedor Central

El contenedor central basado en Fedora cumple la función de nodo de supervisión en esta infraestructura híbrida [5]. Gracias a la combinación de Grafana, Prometheus y Zabbix, es posible recolectar métricas de rendimiento y analizar tendencias en el uso de recursos [11].

III. PROCEDIMIENTO Y RESULTADOS

A. Instalación de QEMU/KVM y configuración de red en el host

Esta sección detalla el proceso seguido para configurar el entorno de virtualización en el equipo anfitrión. Se instalaron herramientas como QEMU, KVM, libvirt y Virt-Manager, necesarias para el manejo de máquinas virtuales con soporte de aceleración por hardware.

1) Instalación de herramientas de virtualización

En la Figura 1 se puede observar la ejecución del comando que actualiza los repositorios e instala los paquetes necesarios para la virtualización en Linux.

```
sudo apt update && sudo apt install
    qemu-kvm libvirt-daemon-system
    libvirt-clients bridge-utils
    virt-manager -y
```

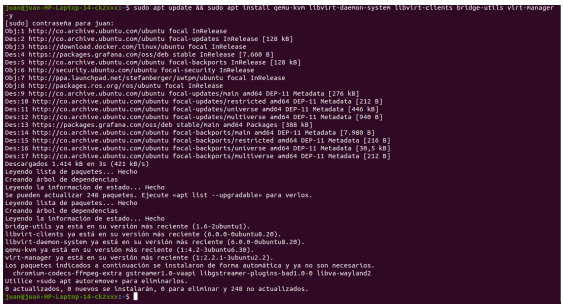

Figura 1: En la imagen se puede observar la instalación del entorno de virtualización mediante terminal. Fuente: Elaboración propia.

2) Verificación de soporte KVM

En la Figura 2 se observa el resultado de ejecutar 'kvm-ok', que indica si el sistema soporta virtualización por hardware.

```
kvm-ok
```

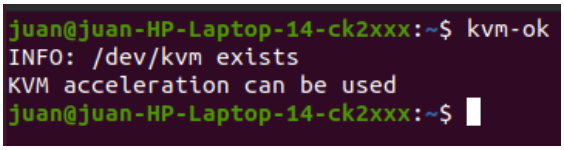

Figura 2: Verificación de soporte KVM. En la imagen se puede observar que el sistema soporta aceleración por hardware. Fuente: Elaboración propia.

3) Asignación de permisos al usuario

Para que el usuario pueda gestionar máquinas virtuales sin errores de permisos, fue agregado a los grupos 'libvirt' y 'kvm', como se muestra en la Figura 3.

```
sudo usermod -aG libvirt $(
    whoami)
sudo usermod -aG kvm $(whoami)
```

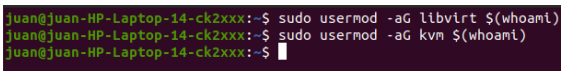

Figura 3: Asignación del usuario a los grupos libvirt y kvm. Fuente: Elaboración propia.

4) Inicio del gestor gráfico Virt-Manager

Con todo configurado, se ejecutó el entorno gráfico para administrar las máquinas virtuales, como se muestra en la Figura 4.

```
virt-manager
```

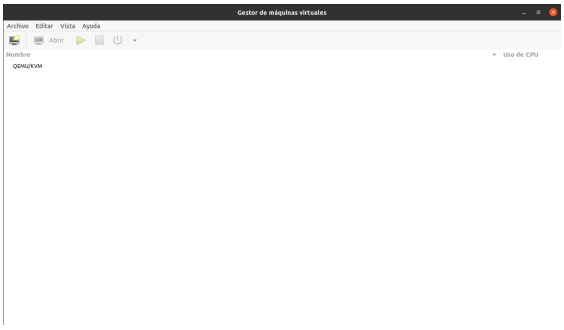

Figura 4: Virt-Manager abierto. En la imagen se puede observar el entorno gráfico listo para gestionar máquinas virtuales. Fuente: Elaboración propia.

B. Configuración de subred tipo bridge para las máquinas virtuales

Se configuró una subred tipo puente para permitir conectividad directa de las máquinas virtuales a la red física.

1) Edición del archivo Netplan

El archivo de red fue abierto con el editor 'nano', como se muestra en la Figura 5.

```
sudo nano /etc/netplan/01-
    network-manager-all.
    yaml
```

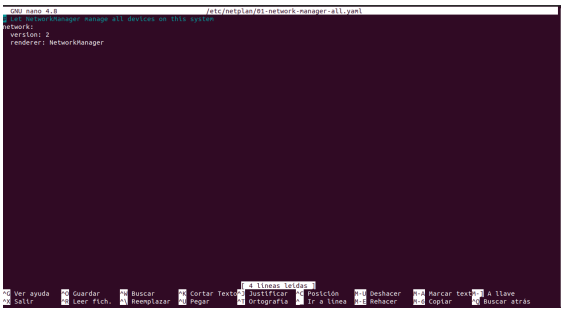

Figura 5: Edición del archivo Netplan. Fuente: Elaboración propia.

2) Contenido del archivo de configuración

El contenido define una interfaz ‘br0‘ puenteada a ‘enp3s0‘, como se ve en la Figura 6.

```
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp3s0:
      dhcp4: no
  bridges:
    br0:
      interfaces: [enp3s0]
      dhcp4: yes
      parameters:
        stp: false
        forward-delay: 0
```

Figura 6: Configuración de subred tipo bridge. Fuente: Elaboración propia.

3) Aplicación de la configuración

Luego de guardar los cambios, se aplicaron con el comando ‘netplan apply‘.

```
sudo netplan apply
```

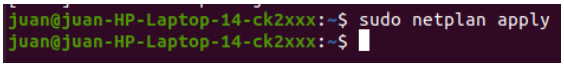

Figura 7: Aplicación de los cambios de red. Fuente: Elaboración propia.

4) Verificación de interfaces activas

Finalmente, se verificó la creación de ‘br0‘ con el comando ‘ip a‘.

```
ip a
```

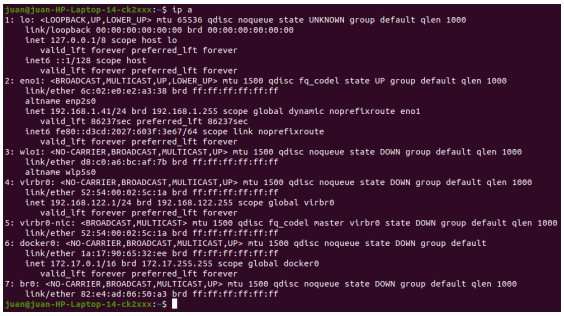

Figura 8: Visualización de interfaces activas. Fuente: Elaboración propia.

C. Descarga e instalación de sistemas operativos en máquinas virtuales

En esta sección se detalla el proceso para crear discos virtuales e instalar tres distribuciones de Linux: Rocky, Manjaro y Arch, usando QEMU/KVM. Se optó por utilizar el doble del espacio en disco normalmente recomendado para cada sistema operativo con el fin de permitir una instalación fluida y suficiente margen para pruebas posteriores.

1) Creación de discos virtuales

El primer paso consistió en crear el directorio `/VMs` y generar los discos virtuales en formato `qcow2`. Este formato es eficiente para entornos virtualizados debido a su capacidad de asignación dinámica de espacio. Se crearon discos de 30 GB, 20 GB y 10 GB para Rocky, Manjaro y Arch respectivamente, aproximadamente el doble de lo que normalmente requiere cada distribución para su instalación mínima.

```
mkdir -p ~/VMs

qemu-img create -f qcow2 ~/VMs/
    rocky.qcow2 30G
qemu-img create -f qcow2 ~/VMs/
    manjaro.qcow2 20G
qemu-img create -f qcow2 ~/VMs/
    arch.qcow2 10G
```

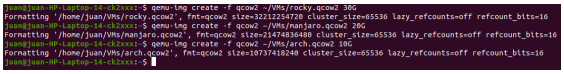

Figura 9: En la imagen se puede observar la creación de discos virtuales para Rocky, Manjaro y Arch. Fuente: Elaboración propia.

2) Instalación de Rocky Linux

Para instalar Rocky Linux se utilizó el comando `virt-install` con los parámetros necesarios: nombre de la VM, memoria (2 GB), número de CPUs (2), medio de instalación (ISO), disco, red (puente br0), tipo de visualización

(VNC), y opciones de arranque sin consola automática. Estos parámetros aseguran una instalación controlada y optimizada. En la Figura 10 se muestra este proceso.

```
sudo virt-install \
--name=Rocky_VM \
--ram=2048 \
--vcpus=2 \
--hvm \
--cdrom="$HOME/Sistemas_operativos
    /Rocky.iso" \
--disk path=$HOME/VMs/rocky.qcow2,
    format=qcow2 \
--network bridge=br0 \
--graphics vnc \
--boot useserial=on \
--noautoconsole
```

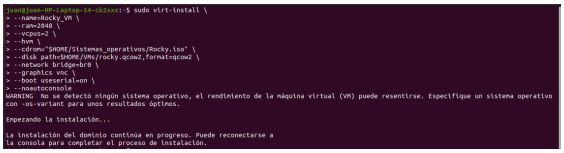

Figura 10: En la imagen se puede observar el proceso de instalación de Rocky Linux en una máquina virtual. Fuente: Elaboración propia.

3) *Instalación de Manjaro Linux*

Para Manjaro se usó una instalación similar, añadiendo el parámetro `--os-variant=archlinux` para optimizar la compatibilidad con esta distribución basada en Arch. Se asignaron los mismos recursos de CPU y memoria que en el caso anterior. En la Figura 11 se observa la ejecución del proceso.

```
sudo virt-install \
--name=Manjaro_VM \
--ram=2048 \
--vcpus=2 \
--os-variant=archlinux \
--hvm \
--cdrom="$HOME/Sistemas_operativos
    /manjaro.iso" \
--disk path=$HOME/VMs/manjaro.
    qcow2,format=qcow2 \
--network bridge=br0 \
--graphics vnc \
--boot useserial=on \
--noautoconsole
```

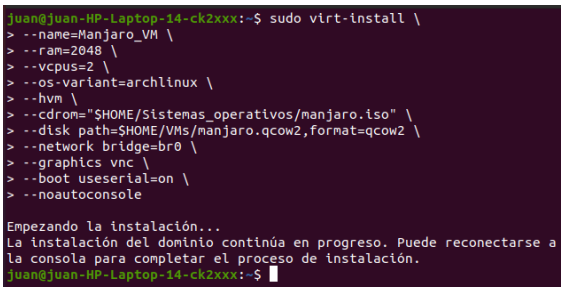

Figura 11: En la imagen se puede observar el proceso de instalación de Manjaro Linux. Fuente: Elaboración propia.

4) *Instalación de Arch Linux*

Arch Linux se instaló utilizando parámetros similares a los de Manjaro, ya que ambas comparten una base común. Se definió el ISO de instalación, el disco virtual previamente creado y se configuró el entorno gráfico con VNC para monitorear la instalación desde la interfaz. En la Figura 12 se observa la ejecución del comando correspondiente.

```
sudo virt-install \
--name=Arch_VM \
--ram=2048 \
--vcpus=2 \
--os-variant=archlinux \
--hvm \
--cdrom="$HOME/Sistemas_operativos
    /archlinux.iso" \
--disk path=$HOME/VMs/arch.qcow2,
    format=qcow2 \
--network bridge=br0 \
--graphics vnc \
--boot useserial=on \
--noautoconsole
```

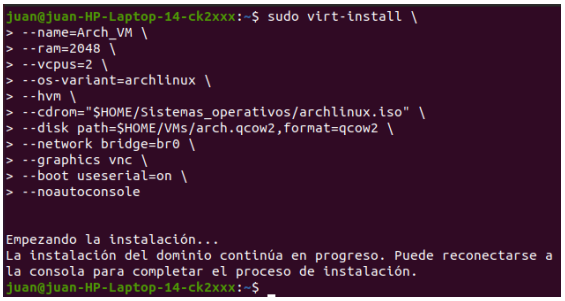

Figura 12: En la imagen se puede observar el inicio del proceso de instalación de Arch Linux. Fuente: Elaboración propia.

5) *Verificación de las máquinas virtuales creadas*

Finalmente, se accedió a Virt-Manager para verificar que las máquinas virtuales Rocky, Manjaro y Arch fueron creadas correctamente. En la Figura 13 se observa la interfaz gráfica mostrando las VMs ya registradas y listas para ser utilizadas.

```
virt-manager
```

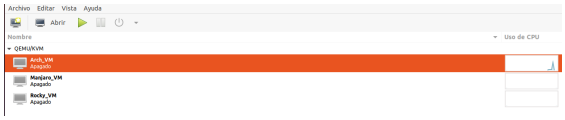

Figura 13: Visualización de las máquinas virtuales Rocky, Manjaro y Arch creadas exitosamente. Fuente: Elaboración propia.

D. Instalación y configuración de Rocky Linux

Una vez finalizada la instalación de Rocky Linux en la máquina virtual, se procedió con su configuración básica y posterior instalación de Docker para gestionar contenedores. Esta distribución se eligió por su compatibilidad con entornos empresariales y su estabilidad.

1) Instalación gráfica del sistema

En la Figura 14 se puede observar la interfaz gráfica de instalación de Rocky Linux. El proceso se realizó en modo GUI, lo que facilitó la selección de parámetros regionales y configuración de red.

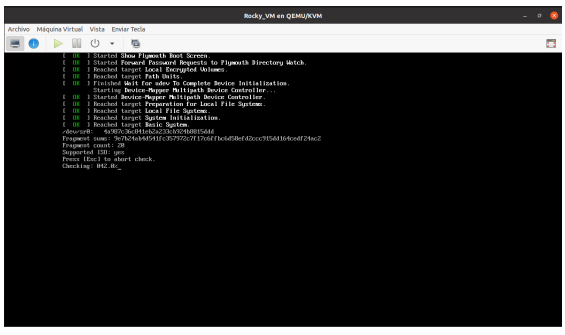

Figura 14: Instalación gráfica de Rocky Linux iniciada desde Virt-Manager. Fuente: Elaboración propia.

2) Configuración inicial del sistema

La Figura 15 muestra la ventana de configuración donde se establecieron aspectos como la zona horaria, idioma del sistema, y se asignaron disco y usuarios. Es importante destacar que durante esta configuración se detectaron advertencias por ausencia de contraseña root y creación de usuario, lo cual fue corregido posteriormente.

Figura 15: Configuración inicial de Rocky Linux: selección de idioma, disco, red y creación de usuarios. Fuente: Elaboración propia.

3) Descarga e instalación del sistema

En la Figura 16 se observa el proceso de instalación de paquetes del sistema, específicamente los módulos del kernel. Esta instalación fue completada satisfactoriamente y sin errores.

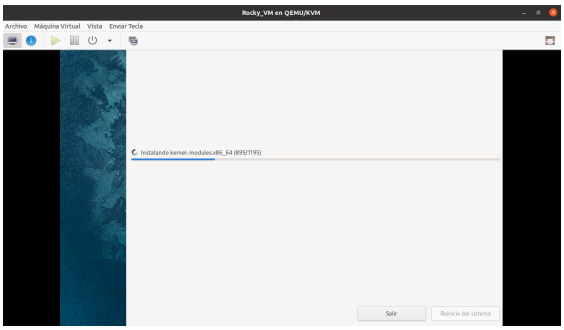

Figura 16: Proceso de instalación del sistema Rocky Linux. Fuente: Elaboración propia.

4) Verificación del arranque del sistema

Después de finalizar la instalación y reiniciar el sistema, se realizó una verificación automática del medio ISO como se muestra en la Figura 17. Esta verificación confirmó que el medio era válido y que el sistema podía iniciar correctamente.

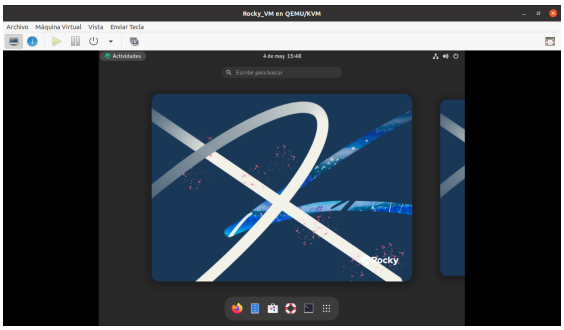

Figura 17: Verificación del medio de instalación durante el primer arranque de Rocky Linux. Fuente: Elaboración propia.

5) *Instalación de Docker en Rocky Linux*

Para instalar Docker, se añadió el repositorio oficial y se utilizaron los comandos mostrados en el Código III-D5. En la Figura 18 se observa la descarga e instalación de los paquetes necesarios: ‘docker-ce‘, ‘docker-ce-cli‘, ‘containerd.io‘ y plugins adicionales como ‘docker-compose‘.

```
dnf config-manager --add-repo https
    ://download.docker.com/linux/
    centos/docker-ce.repo

dnf install docker-ce docker-ce-cli
    containerd.io -y

systemctl enable --now docker
```

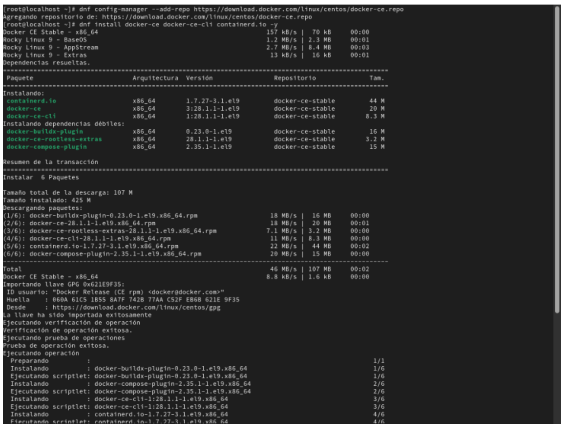

Figura 18: Instalación de Docker CE y sus componentes en Rocky Linux. Fuente: Elaboración propia.

6) *Verificación de instalación de Docker*

Finalmente, se ejecutó el comando `docker version` para confirmar que tanto el cliente como el servidor de Docker estaban correctamente instalados. Como se muestra en la Figura 19, se obtuvo la versión 28.1.1 del motor de Docker junto con los detalles del contenedor y entorno del sistema.

```
docker version
```

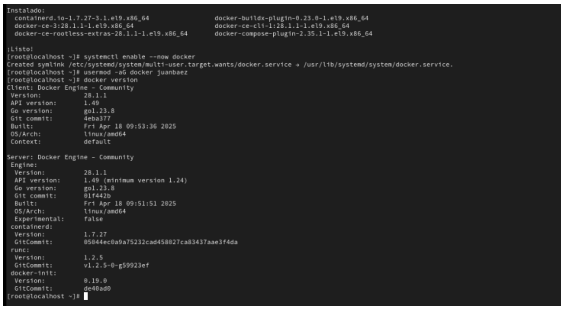

Figura 19: Verificación de la instalación de Docker en Rocky Linux. Fuente: Elaboración propia.

E. *Instalacion de Manjaro*

1) *Instalación de Manjaro Linux*

En esta etapa se procedió con la instalación gráfica de Manjaro Linux 25.0.0 “Zetar” en la máquina virtual configurada. Durante el proceso, se seleccionaron las configuraciones de ubicación, teclado, particionado automático del disco y opciones predeterminadas del sistema. En la Figura 20 se observa el asistente gráfico en plena ejecución de la instalación del sistema.

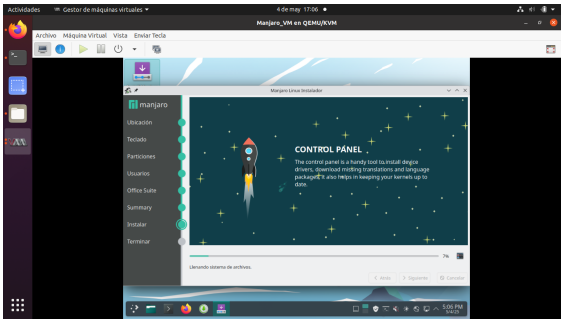

Figura 20: Proceso de instalación gráfica de Manjaro Linux. Fuente: Elaboración propia.

2) *Configuración de particiones y opciones finales*

Durante la instalación, se optó por borrar completamente el disco virtual y utilizar el sistema de archivos `btrfs`. En la Figura 21 se puede observar cómo el sistema detecta 20,GB

sin particionar, y la configuración resultante asigna todo el espacio a la partición raíz del sistema Manjaro.

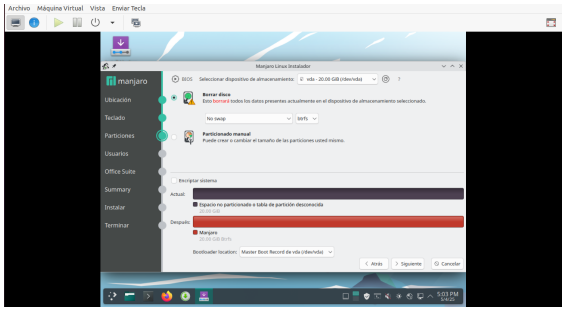

Figura 21: Configuración automática del particionado del disco en la instalación de Manjaro. Fuente: Elaboración propia.

3) *Finalización de la instalación*

Una vez finalizada la descarga e instalación de los paquetes, el sistema mostró un mensaje de confirmación indicando que la instalación se había completado exitosamente. En la Figura 22 se observa el mensaje "Listo" del instalador, que sugiere reiniciar o continuar usando el entorno Live.

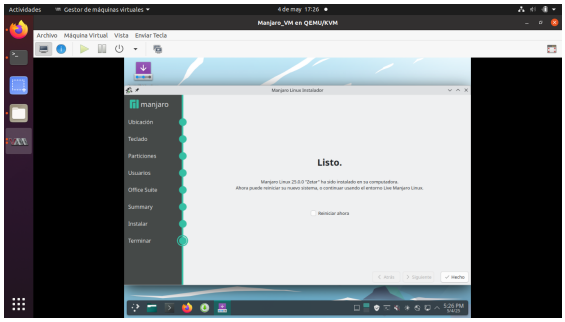

Figura 22: Finalización del proceso de instalación gráfica de Manjaro. Fuente: Elaboración propia.

4) *Instalación de Docker*

La instalación de Docker en Manjaro se realizó utilizando el gestor de paquetes `pacman`, con la opción `--noconfirm` para evitar interrupciones. Posteriormente, se activó el servicio de Docker y se otorgaron los permisos necesarios al usuario actual para gestionar contenedores sin privilegios de superusuario. En la Figura 23 se muestra este proceso desde la terminal.

```
sudo pacman -S docker --noconfirm
sudo systemctl enable --now docker
sudo usermod -aG docker $(whoami)
newgrp docker
```

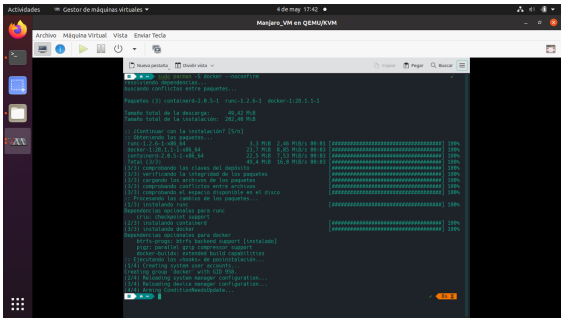

Figura 23: Instalación y configuración de Docker en Manjaro. Fuente: Elaboración propia.

5) *Verificación de Docker*

Finalmente, se verificó la instalación de Docker mediante el comando `docker version`, el cual muestra la versión del cliente y del servidor, así como sus respectivas dependencias. En la Figura 24 se puede observar que Docker fue instalado y configurado correctamente en Manjaro.

```
docker version
```

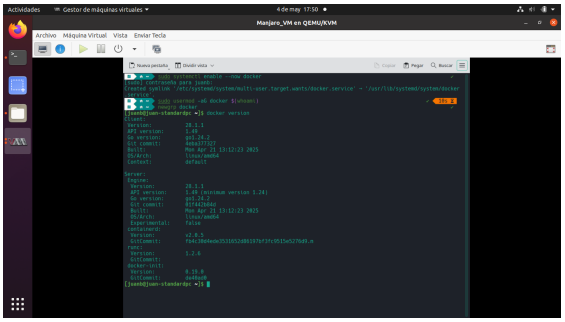

Figura 24: Verificación de la instalación y versión de Docker en Manjaro. Fuente: Elaboración propia.

F. *Intalación de ArchLinux*

1) *Intento de instalación de Arch Linux*

El proceso de instalación de Arch Linux presentó múltiples dificultades que impidieron completarlo con éxito, a diferencia de las instalaciones previas de Rocky y Manjaro. A continuación, se documenta el procedimiento seguido y los errores que surgieron durante el intento.

En la Figura 25 se puede observar el inicio del proceso de instalación de Arch Linux, donde se solicitó seleccionar el tipo de etiqueta para el disco, incluyendo opciones como gpt, dos, sgi y sun.

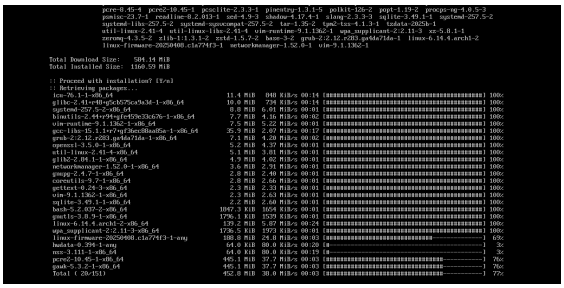

Figura 25: Inicio de instalación de Arch Linux: selección del tipo de etiqueta de partición. Fuente: Elaboración propia.

Posteriormente, se continuó con la descarga de los paquetes esenciales para el sistema. En la Figura 26 se visualiza la instalación de los componentes del sistema base, incluyendo el kernel, utilidades del sistema y herramientas de red. Aunque esta etapa se ejecutó correctamente, evidenció el alto volumen de paquetes requeridos, con un tamaño total de descarga de más de 500 MB y una instalación que superó los 1100 MB.

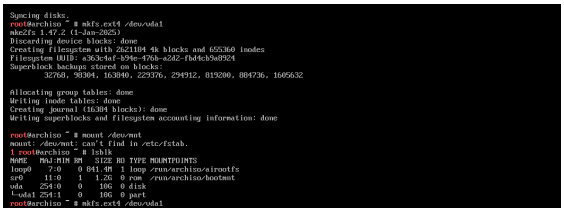

Figura 26: Descarga e instalación de paquetes base en Arch Linux. Fuente: Elaboración propia.

Durante la fase de creación de particiones, se ejecutaron los comandos para formatear el disco y crear el sistema de archivos. Sin embargo, en la Figura 27 se puede observar que al intentar montar el punto `/mnt`, el sistema arrojó un error: `can't find in /etc/fstab`, lo cual detuvo el avance del proceso. Este error suele aparecer cuando el sistema de archivos no se ha definido o montado correctamente, o cuando las particiones no han sido creadas y asignadas adecuadamente.

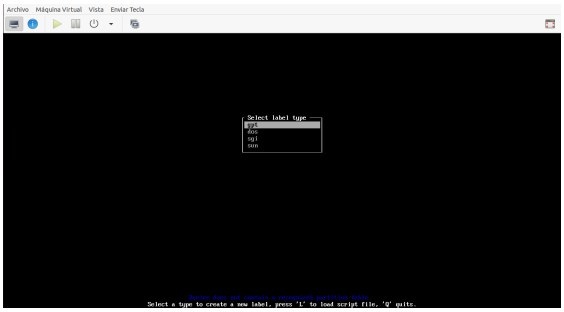

Figura 27: Error durante la creación de particiones y montaje del sistema en Arch Linux. Fuente: Elaboración propia.

Debido a este inconveniente técnico y la complejidad de la instalación manual de Arch, que requiere la ejecución secuencial y correcta de múltiples comandos sin entorno gráfico de apoyo, no fue posible finalizar la instalación de este sistema operativo en la máquina virtual. Este fallo resalta la diferencia entre distribuciones amigables como Manjaro y entornos avanzados como Arch, donde el usuario asume mayor responsabilidad sobre cada aspecto del sistema.

G. Verificación y despliegue de contenedores Docker en Rocky y Manjaro

1) Prueba de ejecución del contenedor `hello-world`

Para verificar que Docker se instaló correctamente en ambos sistemas operativos, se ejecutó el contenedor `hello-world`. En la Figura 28 se muestra la salida del comando en Manjaro, y en la Figura 29 la salida correspondiente en Rocky. En ambos casos, el mensaje confirma que el demonio Docker fue contactado, la imagen fue descargada y el contenedor ejecutado exitosamente.

```
docker run hello-world
```

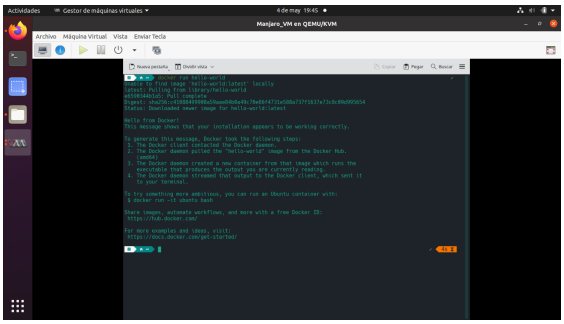

Figura 28: Verificación de Docker en Manjaro usando el contenedor `hello-world`. Fuente: Elaboración propia.

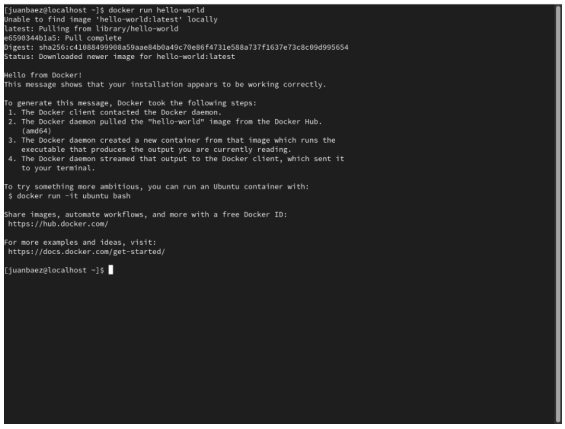

Figura 29: Verificación de Docker en Rocky con el contenedor `hello-world`. Fuente: Elaboración propia.

2) *Despliegue de contenedores para la plataforma Zabbix*

Una vez verificado el funcionamiento de Docker, se intentó desplegar contenedores para la plataforma de monitoreo Zabbix, junto con una base de datos MySQL. En la Figura 30 se observa la ejecución de los comandos para levantar los servicios utilizando la red personalizada `zabbix-net`. Inicialmente se desplegaron dos contenedores, uno para el servidor MySQL y otro para la interfaz web de Zabbix.

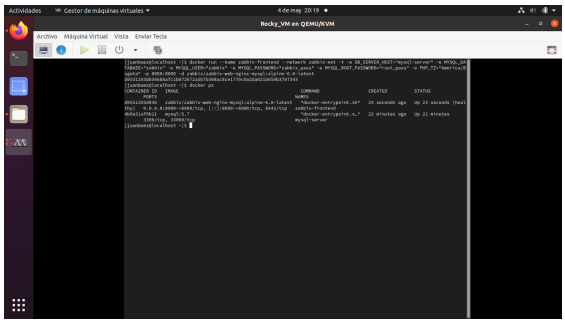

Figura 30: Ejecución de contenedores Zabbix y MySQL en Docker. Fuente: Elaboración propia.

3) *Evidencia de conflictos y errores*

Durante el proceso de despliegue se presentaron varios errores. En la Figura 31 se observa que al intentar ejecutar el contenedor `mysql-server` nuevamente, Docker arroja un error de conflicto indicando que el nombre ya estaba en uso. Esto sucedió por un intento previo fallido que dejó el contenedor activo, dificultando la reutilización del nombre.

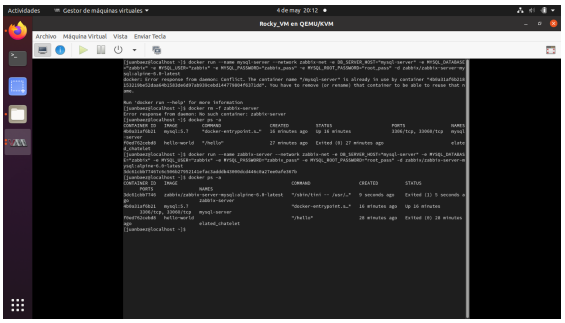

Figura 31: Intento de despliegue de contenedores y errores de conflicto de nombres. Fuente: Elaboración propia.

4) *Reinstalación del contenedor de base de datos*

Posteriormente, se intentó nuevamente la instalación del contenedor MySQL. En la Figura 32 se muestra cómo Docker descarga correctamente la imagen y procede con su despliegue sin errores de sintaxis, logrando activar el contenedor exitosamente.

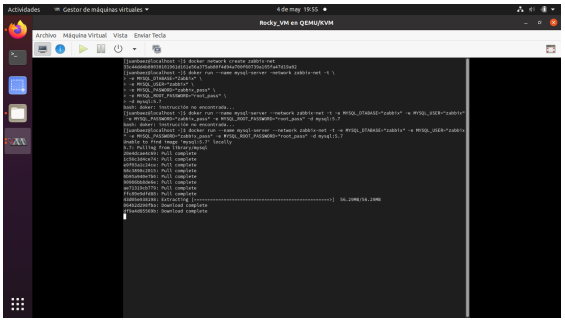

Figura 32: Instalación correcta del contenedor MySQL. Fuente: Elaboración propia.

5) *Error de conexión con la base de datos desde Zabbix*

Finalmente, se accedió a la interfaz web de Zabbix para completar la configuración. Sin embargo, como se observa en la Figura 33, la aplicación arrojó un error de base de datos, indicando que no fue posible establecer conexión con la configuración especificada. Este problema suele estar relacionado con variables de entorno incorrectas o tiempos de espera no respetados en la dependencia entre contenedores.

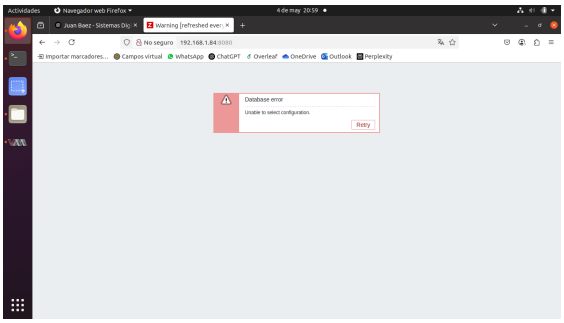

Figura 33: Error de conexión a la base de datos desde la interfaz de Zabbix. Fuente: Elaboración propia.

IV. CONCLUSIONES

Durante el desarrollo de este entorno híbrido de pruebas se pudo comprobar que la combinación de máquinas virtuales y contenedores es una estrategia eficiente para evaluar el comportamiento y desempeño de distintos sistemas operativos. La virtualización mediante QEMU/KVM demostró ser una herramienta robusta para la emulación de sistemas completos sin comprometer el entorno anfitrión. La instalación de Rocky Linux, Manjaro y Arch Linux permitió observar diferencias significativas en la filosofía de configuración y administración de cada distribución.

En el caso de Rocky Linux, se logró realizar una instalación completa con interfaz gráfica, demostrando su orientación a entornos empresariales estables. La instalación de Docker se ejecutó sin inconvenientes y permitió desplegar contenedores, validando la compatibilidad del sistema con arquitecturas modernas de contenedorización.

La experiencia con Manjaro fue igualmente exitosa. Su instalador gráfico facilitó el proceso y permitió trabajar con el sistema en poco tiempo. La instalación y prueba de Docker se realizó sin complicaciones, evidenciando que Manjaro, a pesar de ser una distribución rolling release, puede ser utilizada en entornos de pruebas con contenedores.

En contraste, Arch Linux presentó numerosos retos. A pesar de los intentos de instalación, el proceso falló por errores en la configuración de particiones y montaje de archivos, resaltando la complejidad de esta distribución en comparación con las anteriores. Este resultado permitió concluir que Arch requiere un mayor dominio del sistema y experiencia técnica por parte del usuario.

La configuración de red mediante Netplan y la creación de una interfaz puente (bridge) fue clave para garantizar la conectividad de las máquinas virtuales con la red externa. Esta configuración permitió que los sistemas virtualizados se comportaran como nodos reales dentro de una red local, lo cual fue esencial para las pruebas de Docker y monitoreo.

El despliegue de contenedores en Rocky y Manjaro permitió realizar pruebas prácticas con herramientas como Zabbix y MySQL. Sin embargo, durante estas pruebas se presentaron errores, como conflictos de nombres de contenedores y fallos en la conexión de la base de datos, lo cual evidenció la necesidad de planificar mejor las dependencias y tiempos de arranque entre servicios.

Uno de los aspectos más enriquecedores fue la posibilidad de visualizar estos errores y aprender de ellos en tiempo real, gracias al entorno controlado que ofrecía la virtualización. Esto reforzó la importancia de los logs del sistema, la gestión de procesos y la correcta configuración de variables de entorno.

Otro aprendizaje importante fue la utilidad de Docker como herramienta ligera para la ejecución de aplicaciones. A diferencia de las máquinas virtuales, los contenedores permitieron desplegar servicios de forma inmediata y con menor consumo de recursos, mostrando su gran ventaja para pruebas ágiles y entornos de desarrollo continuo.

El uso de herramientas de monitoreo como Zabbix se propuso como parte del entorno, aunque no se logró una integración completa debido a errores de conexión con la base de datos. A pesar de esto, el intento de implementación permitió entender cómo estas herramientas pueden coordinarse dentro de infraestructuras híbridas.

Durante todo el proyecto se comprobó que la documentación y captura sistemática de cada paso es esencial para identificar errores, corregirlos y tener trazabilidad. Las imágenes capturadas en cada etapa fortalecieron el informe y permitieron mostrar con claridad el avance del proyecto.

También se concluyó que el uso de terminal fue predominante, tanto en la configuración de red como en la instalación de sistemas y servicios. Esto resalta la importancia de dominar comandos y entornos de línea para administrar infraestructuras virtualizadas.

La experiencia adquirida en la creación y gestión de discos virtuales, configuración de sistemas de archivos, asignación de recursos y uso de gestores gráficos como Virt-Manager aportó una base sólida para futuras implementaciones de entornos de laboratorio o producción.

A nivel técnico, el proyecto permitió evidenciar la diferencia entre sistemas operativos orientados a usuarios (como Manjaro), aquellos enfocados a servidores (como Rocky) y otros más complejos y minimalistas como Arch. Esta diversidad enriqueció la experiencia comparativa y permitió entender qué tipo de sistema es más adecuado para cada escenario.

Finalmente, la integración de contenedores y máquinas virtuales en un mismo entorno de pruebas mostró que ambas tecnologías pueden convivir y complementarse. Mientras las máquinas virtuales ofrecen aislamiento completo y flexibilidad para probar sistemas operativos enteros, los contenedores proporcionan eficiencia y rapidez para desplegar servicios específicos.

En suma, este proyecto cumplió el objetivo de construir un entorno de pruebas funcional, documentado, comparativo y educativo, permitiendo adquirir habilidades clave en virtualización, contenedorización, administración de sistemas Linux y gestión de infraestructura de red.

REFERENCIAS

[1] A. S. Tanenbaum, *Modern Operating Systems*. Pearson, 2015.
[2] D. Merkel, "Docker: Lightweight Linux Containers for Consistent Development and Deployment," *Linux Journal*, vol. 2014, no. 239, 2014.
[3] M. Martin, "System Logging and Monitoring with journalctl," *SysAdmin Magazine*, 2017.
[4] J. Fonseca, *Monitoring IT Infrastructures with Netdata*. Packt Publishing, 2018.
[5] T. Chacon, "Grafana: Real-time Monitoring and Visualization," *IEEE IT Journal*, 2020.
[6] G. Stewart, "Cloud Monitoring with Prometheus and Grafana," *ACM SIGOPS*, vol. 19, no. 2, 2019.
[7] R. Sharma, *Containerization and Kubernetes in Modern IT Deployments*. Springer, 2021.
[8] P. Mavridis, "Trends in Virtualization Technologies," *Journal of IT Research*, 2019.
[9] H. Luo, "QEMU and Virtualization Performance," *IEEE Transactions on Computing*, vol. 18, no. 1, 2018.
[10] D. Bernstein, "Containers and the Future of Cloud Computing," *ACM Cloud Computing*, vol. 32, no. 7, 2014.
[11] S. Khan, *Mastering Prometheus and Grafana*. O'Reilly Media, 2020.
[12] A. Gorodov, "Advanced System Monitoring with Zabbix," *IT Operations Journal*, 2013.
[13] M. Kerrisk, *The Linux Programming Interface*. No Starch Press, 2020.
[14] K. Ishida, "Deep Packet Analysis with Wireshark," *Network Security Journal*, 2016.
[15] B. Douglass, "Hybrid IT Architectures and
[16] Rocky Linux, "Descargas de Rocky Linux," [En línea]. Disponible en: https://rockylinux.org/es-ES/download. [Accedido: 02-may-2025].
[17] Manjaro Linux, "Descargas de Manjaro Linux," [En línea]. Disponible en: https://manjaro.org/products/download/x86. [Accedido: 02-may-2025].
[18] Arch Linux, "Última versión de Arch Linux," [En línea]. Disponible en: https://archlinux.org/releng/releases/2025.04.01/. [Accedido: 02-may-2025].