Proyecto Segundo Corte

Entorno híbrido con QEMU y Docker

Melanie Aponte
Universidad Santo Tomás

29 de Abril

Índice

1 Instalación de Máquinas Virtuales (QEMU)

1.1 Sistemas operativos instalados

- Rocky Linux
- Manjaro Linux
- Arch Linux

1.2 Instalación de Arch Linux en Máquina Virtual con Entorno Gráfico KDE Plasma

1.2.1. Creación de la máquina virtual

- Se creó una nueva máquina virtual en Virt-Manager:
  - Firmware: **UEFI (OVMF)**.
  - Chipset: **Q35**.
  - Disco duro de 20 GB en formato `qcow2`.
  - Memoria RAM de 2 GB o más, y 2 CPU.
  - Fuente de arranque: ISO de Arch Linux oficial.
- Se verificó el arranque en modo UEFI mediante:

```
ls /sys/firmware/efi/efivars
```

1.2.2. Particionado y formateo del disco

- Creación de particiones con `cfdisk` en esquema GPT:
  - 512 MiB EFI System Partition.
  - 2 GiB Swap Partition.
  - Resto para Linux filesystem (raíz).
- Formateo y activación:

```
mkfs.fat -F32 /dev/vda1
mkswap /dev/vda2
swapon /dev/vda2
mkfs.ext4 /dev/vda3
```

1.2.3. Instalación del sistema base

- Montaje de particiones:

```
mount /dev/vda3 /mnt
mkdir /mnt/boot
mkdir /mnt/boot/efi
mount /dev/vda1 /mnt/boot/efi
```

- Instalación de paquetes esenciales:

```
pacstrap /mnt base linux linux-firmware nano networkmanager grub efibootmgr
```

- Generación del archivo `fstab`:

```
genfstab -U /mnt >> /mnt/etc/fstab
```

- Cambio de raíz al nuevo sistema:

```
arch-chroot /mnt
```

1.2.4. Configuraciones iniciales

- Zona horaria:

```
ln -sf /usr/share/zoneinfo/America/Bogota /etc/localtime
hwclock --systohc
```

- Locales:

```
nano /etc/locale.gen
# Descomentar en_US.UTF-8 UTF-8 y es_CO.UTF-8 UTF-8
locale-gen
echo "LANG=en_US.UTF-8" > /etc/locale.conf
```

- Hostname:

```
echo "archvm" > /etc/hostname
nano /etc/hosts
# Agregar:
127.0.0.1    localhost
::1          localhost
127.0.1.1    archvm.localdomain archvm
```

- Instalación y configuración de GRUB:

```
grub-install --target=x86_64-efi --efi-directory=/boot/efi --bootloader-id=Arch
grub-mkconfig -o /boot/grub/grub.cfg
```

- Activación de NetworkManager:

```
systemctl enable NetworkManager
```

1.2.5. Instalación del entorno gráfico KDE Plasma

- Instalación de servidor gráfico y drivers:

  ```
  pacman -S xorg-server xorg-apps xf86-video-qxl
  ```

- Instalación de KDE Plasma y sus aplicaciones:

  ```
  pacman -S plasma kde-applications
  ```

- Instalación y habilitación de SDDM (gestor de sesión gráfica):

  ```
  pacman -S sddm
  systemctl enable sddm
  ```

- Instalación adicional para soporte de sesión:

  ```
  pacman -S xorg-xinit plasma-meta
  ```

1.2.6. Arranque en entorno gráfico

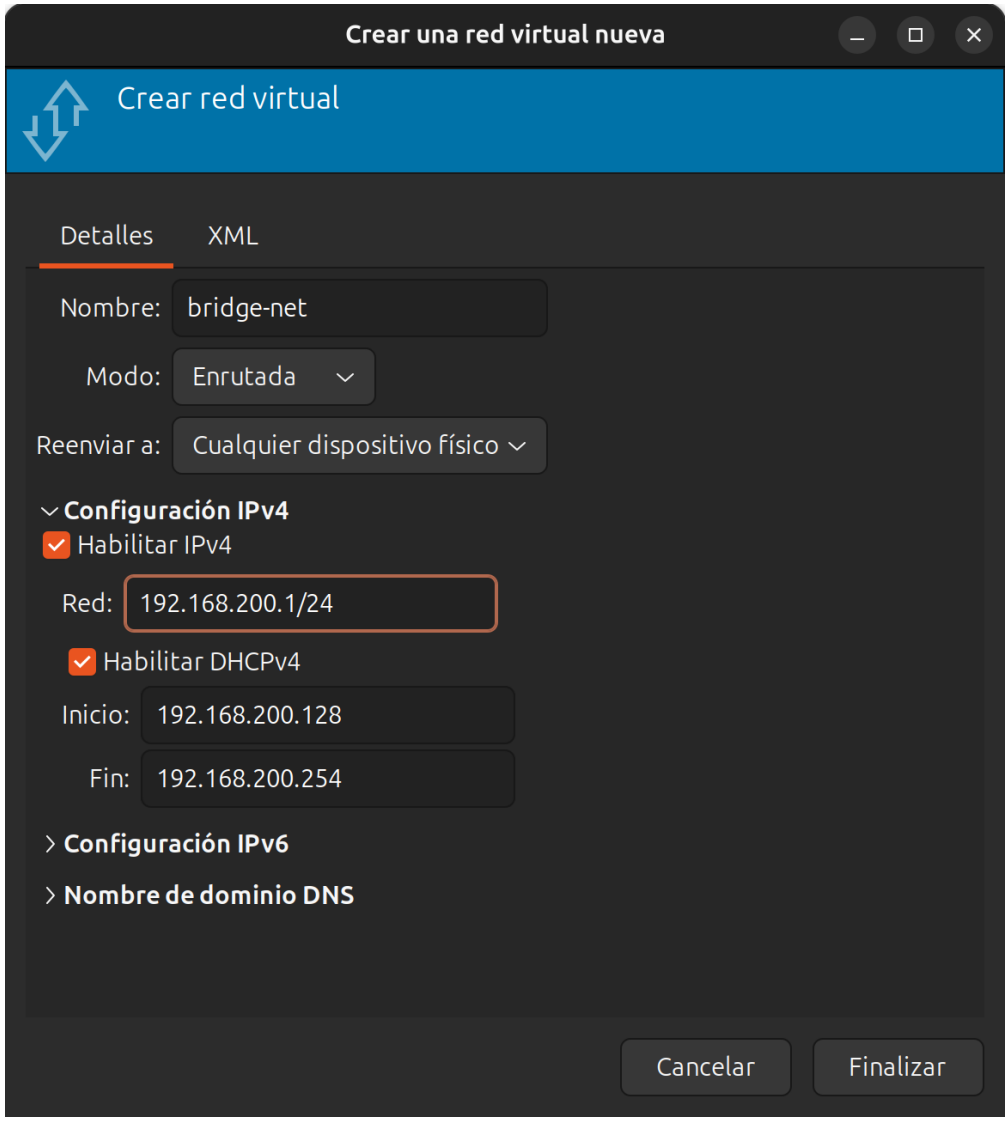
- Tras reiniciar el sistema:

  ```
  reboot
  ```

  Se selecciona la sesión **Plasma (X11)** en SDDM para iniciar correctamente el entorno de escritorio KDE Plasma.

1.3 Configuración de red

- Modo Bridge
- Subred: `192.168.200.0/24`
- Rango DHCP: `192.168.200.128 - 192.168.200.254`



1.4 Herramientas instaladas y pruebas

- Análisis de servicios: `glances`, `bpytop`, `systemd/journalctl`

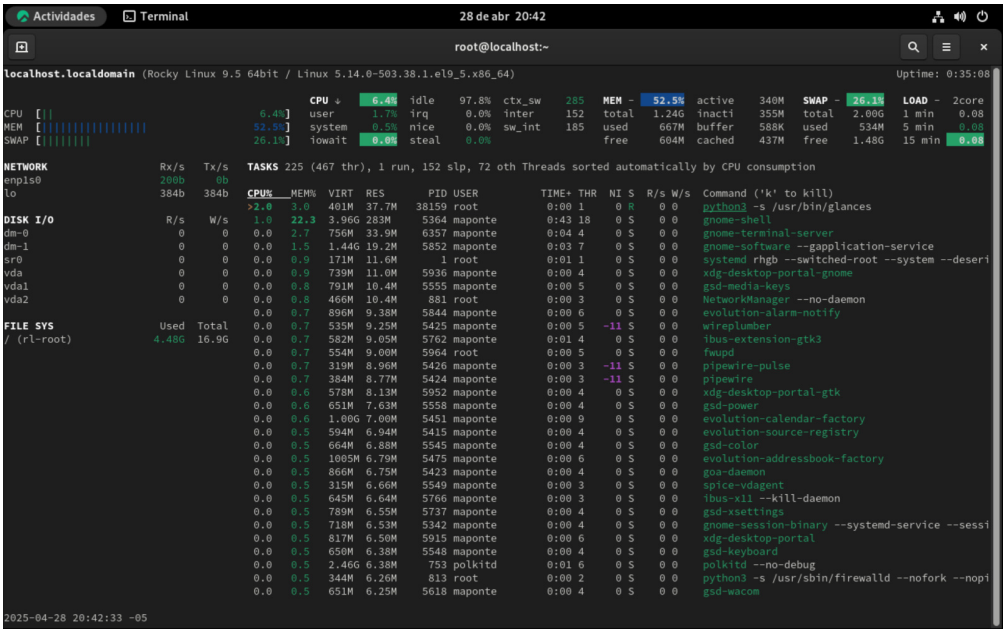


Figura 1: Glances

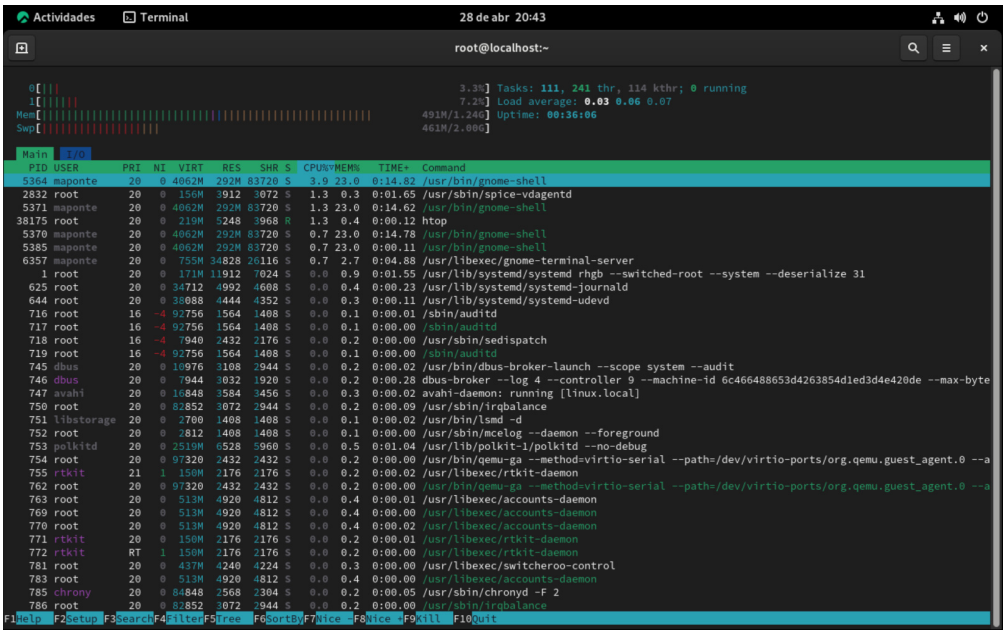


Figura 2: htop

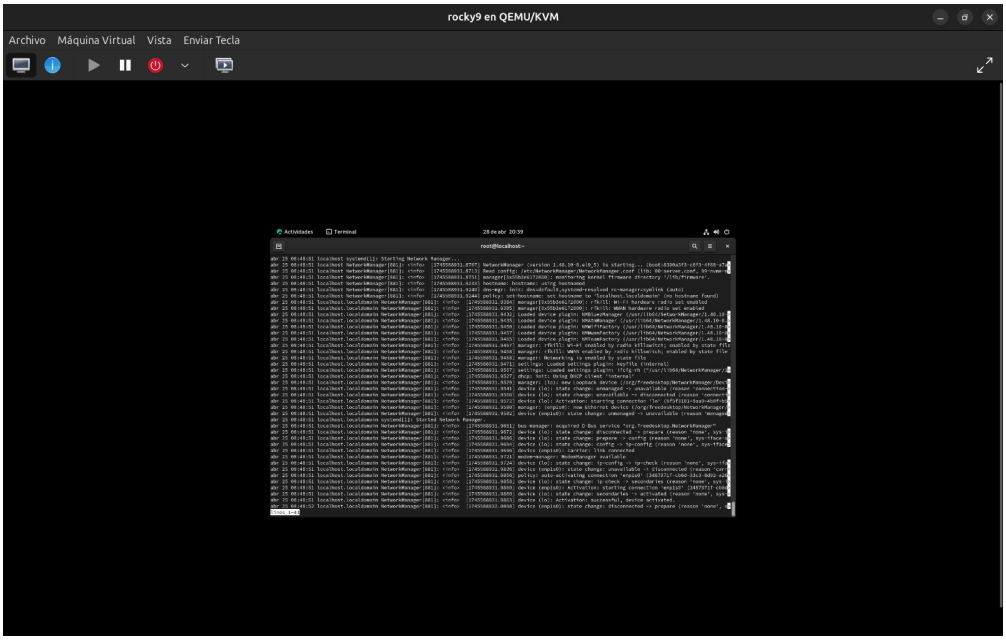


Figura 3: Journalctl

- Análisis de red: `wireshark`, `iftop`, `nethogs`

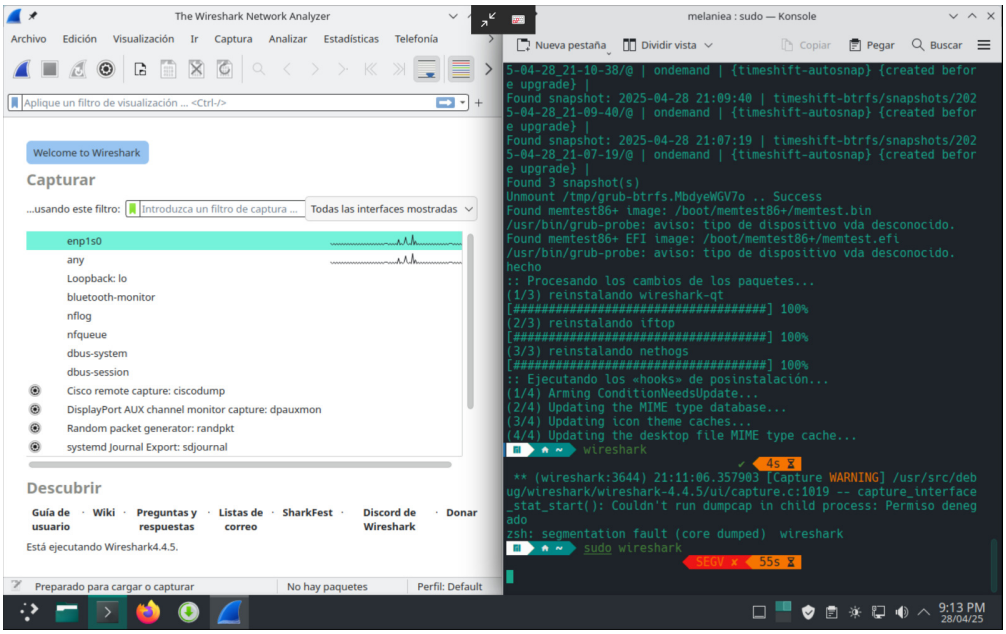


Figura 4: wireshark

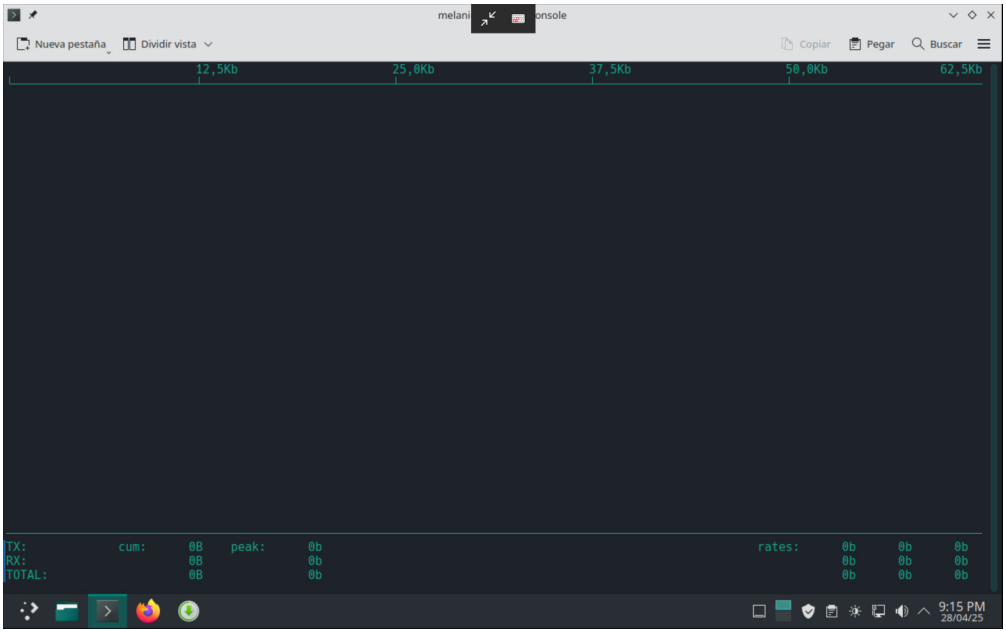


Figura 5: iftop

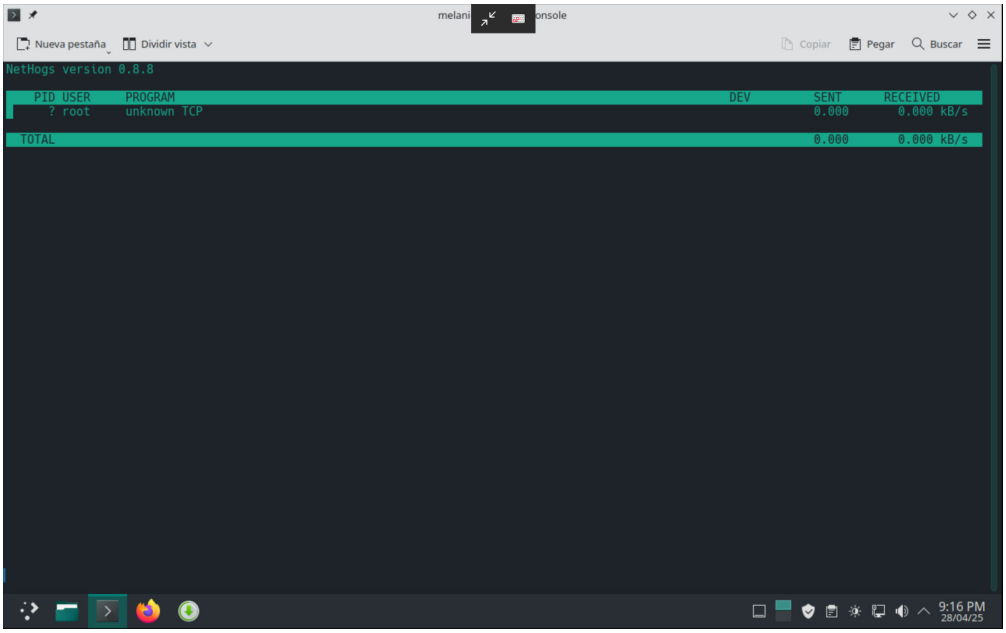


Figura 6: nethogs

- Gestión de archivos: `ncdu`, `baobab`, `tree`, `rsync`

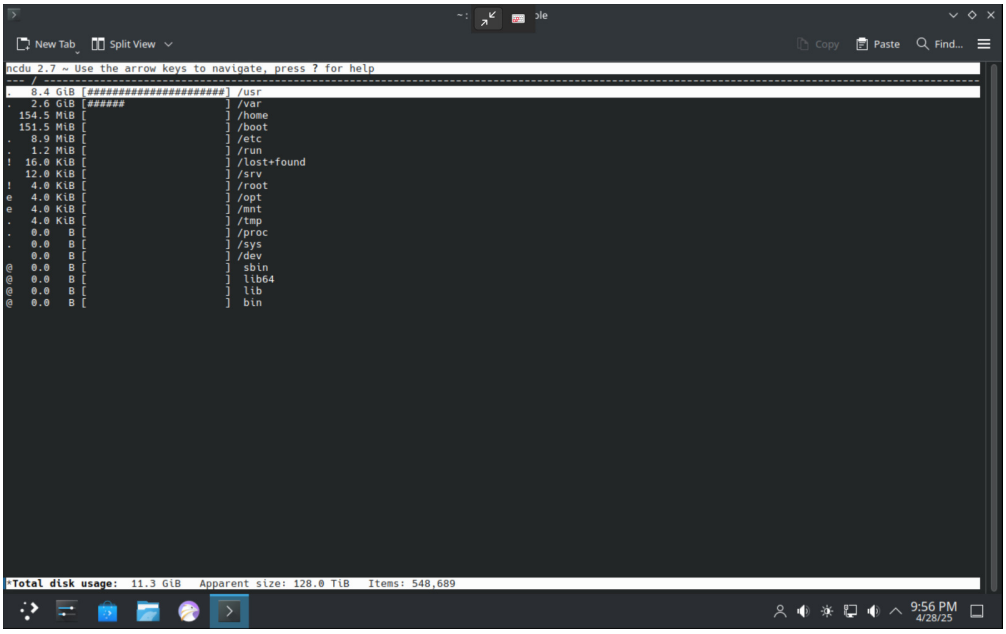


Figura 7: Ncdu

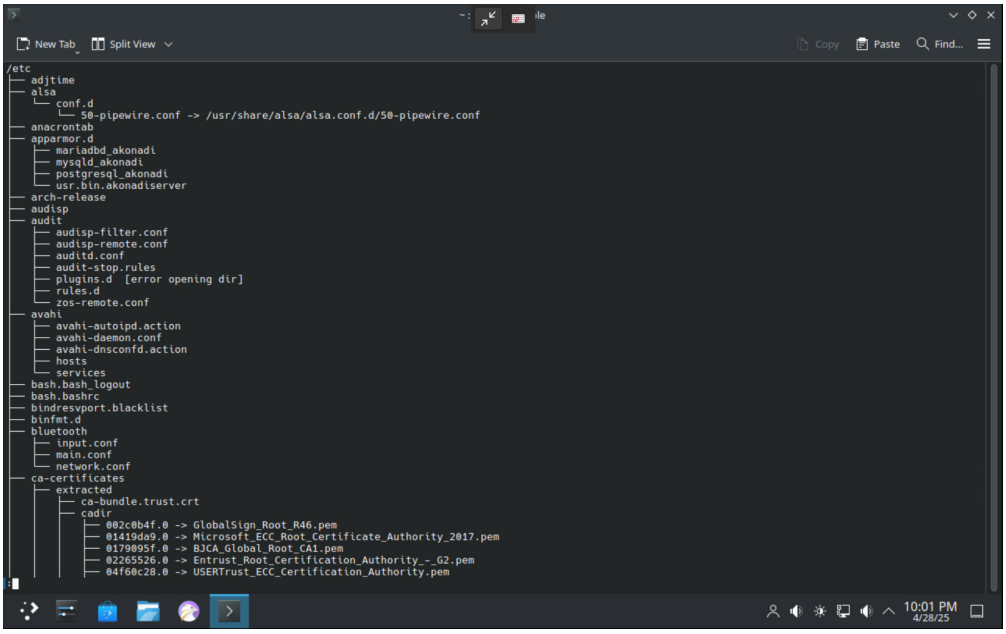


Figura 8: tree

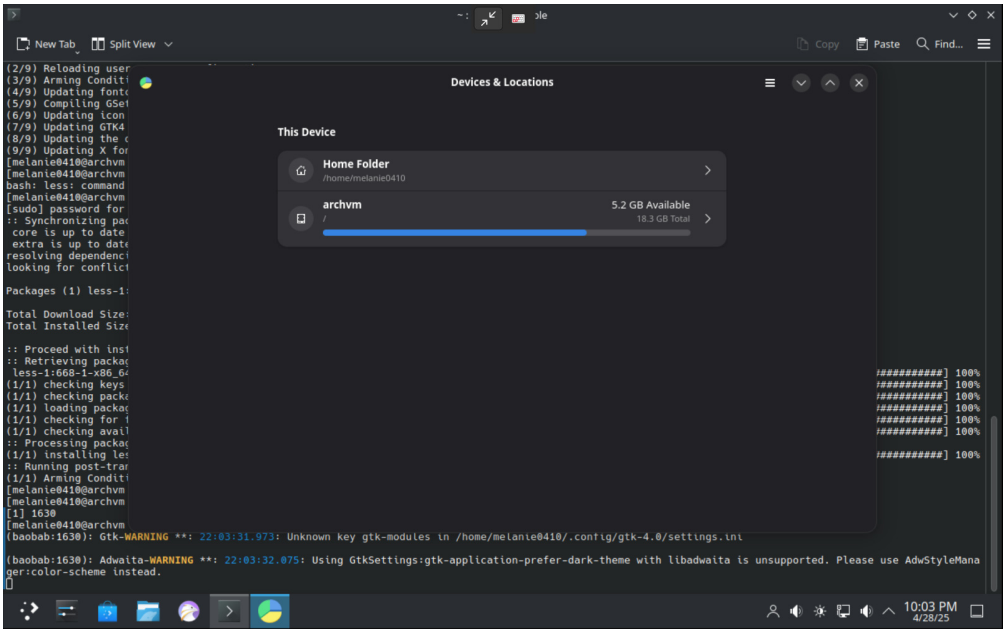


Figura 9: baobab

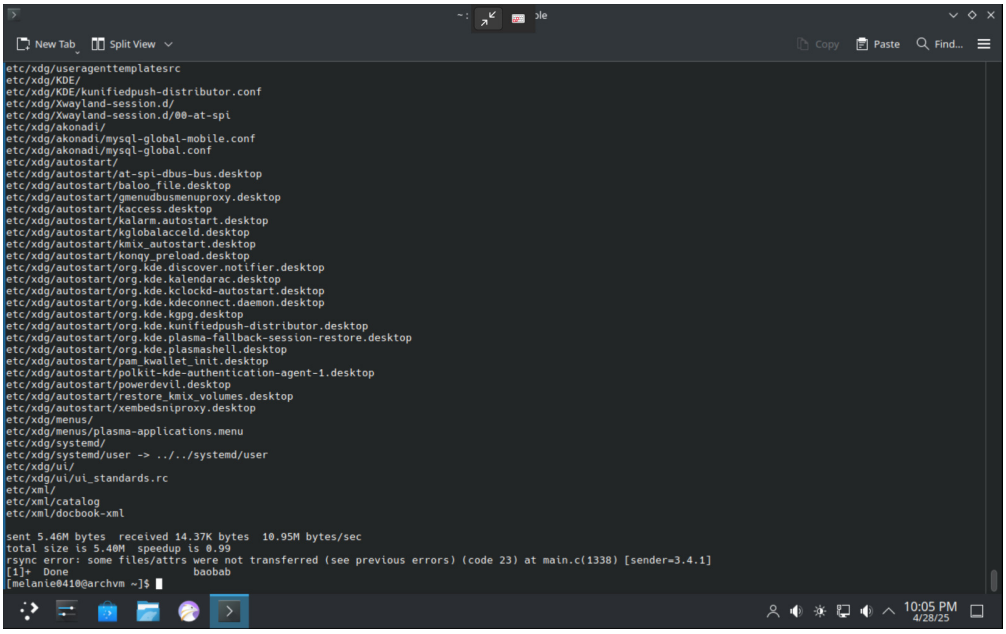


Figura 10: rsync

2 Instalación de Contenedores Docker

2.1 Contenedores usados

- Debian
- Alpine
- Arch (simulando Garuda)

```
d1223dadd942   archlinux                                "bash"
   garuda_container
1a4580ca9c28   debian                                   "bash"
   debian_container
849d84523505   alpine                                   "sh"
   alpine_container
```

2.2 Subred personalizada

- Subred: `172.25.0.0/24`
- Nombre: `red_contenedores`

```
09b7ec109890   red_contenedores                  bridge    local
```

2.3 Herramientas instaladas

- Análisis de hardware: `lshw`, `inxi`, `lsblk`, `udevadm`
- Gestión de logs: `lnav`, `goaccess`
- Visualización de datos: `netdata`, `ctop`, `lazydocker`

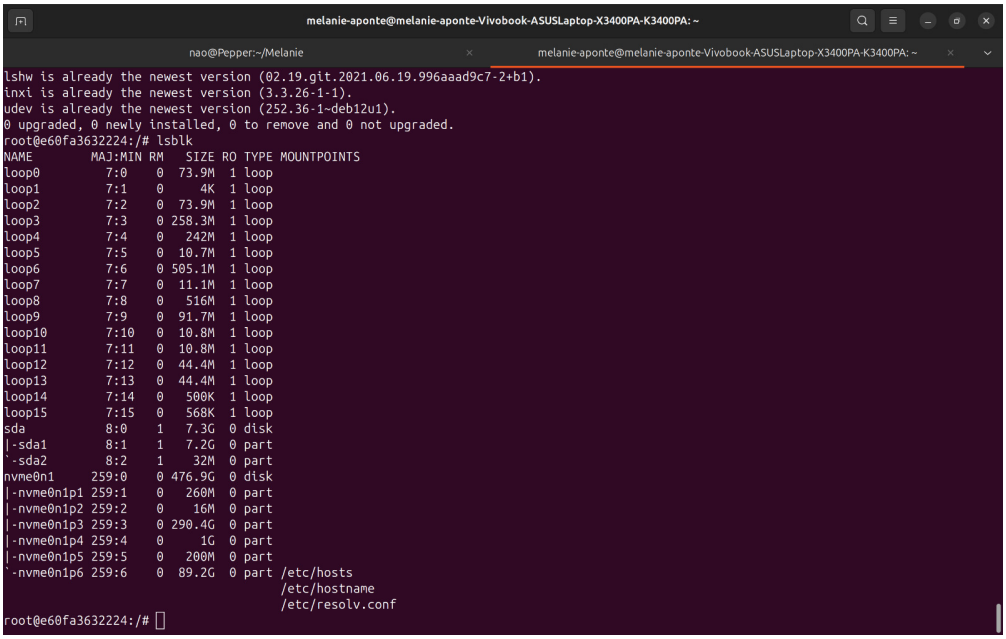


Figura 11: Lsblk

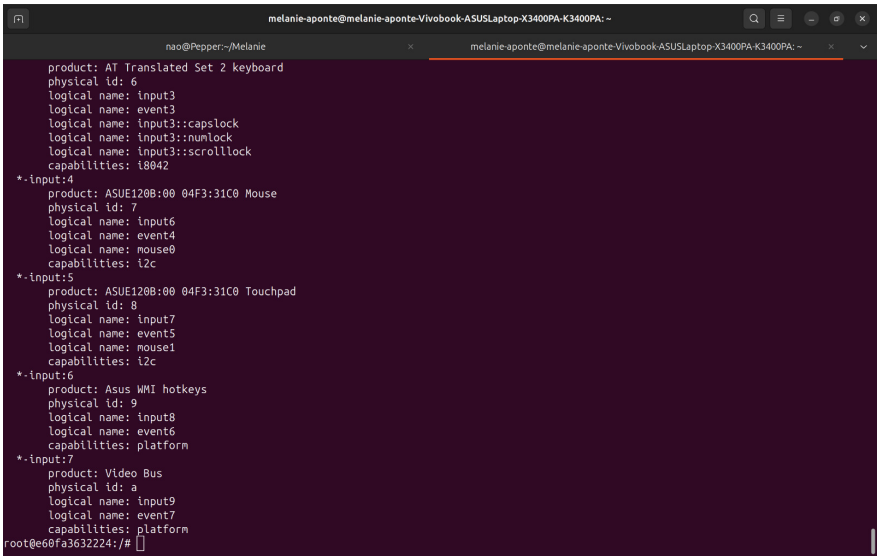


Figura 12: lshw

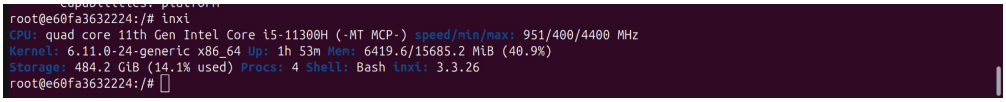


Figura 13: inxi

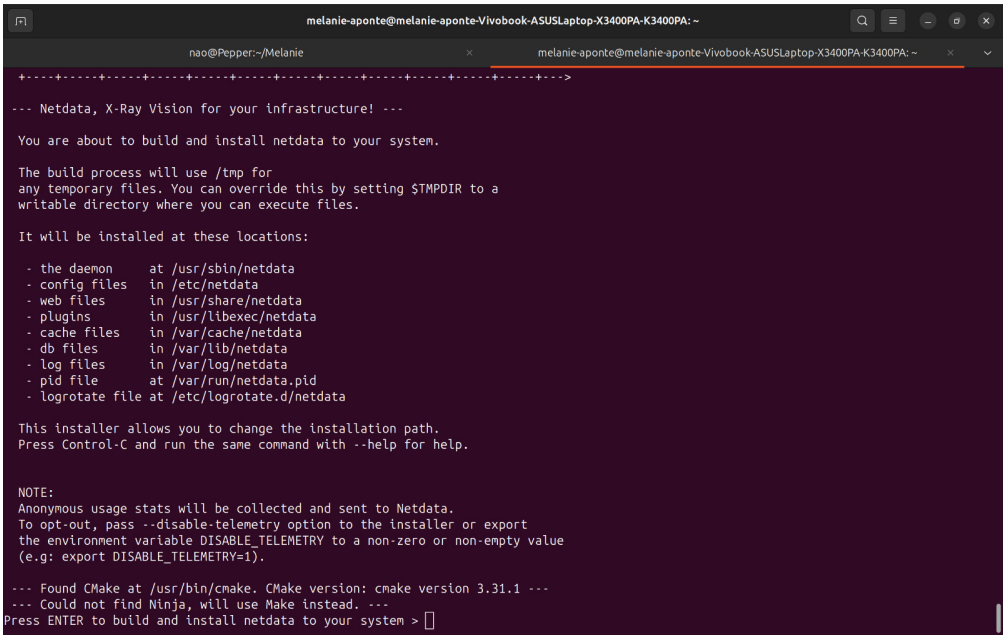


Figura 14: Netdata

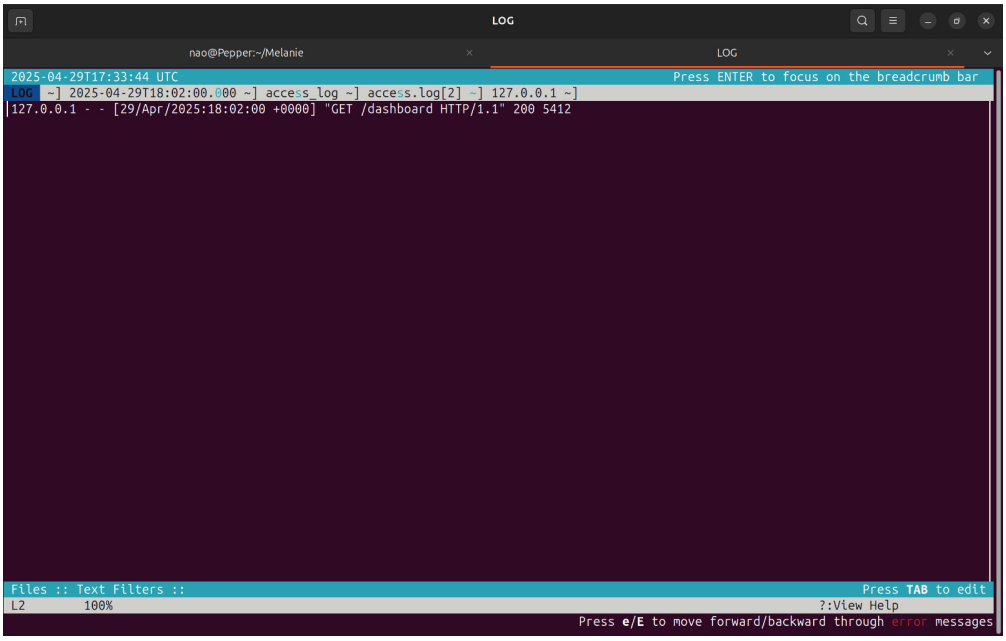


Figura 15: Lnav

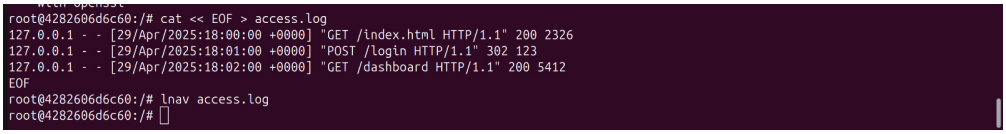


Figura 16: Inav

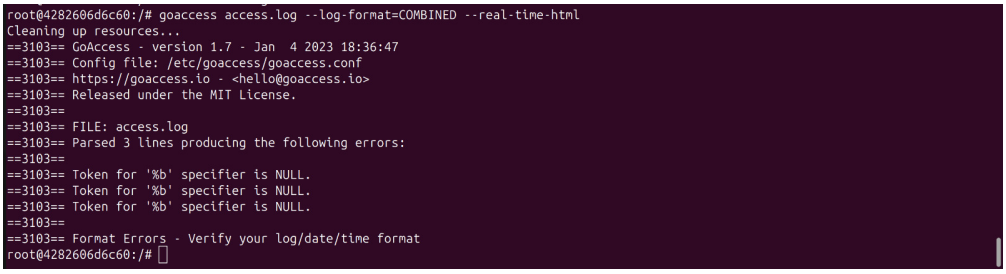


Figura 17: Goaccess

3 Contenedor Central Fedora

3.1 Creación de imagen personalizada

- Herramientas incluidas: `htop`, `net-tools`, `iputils`, `python3`, `git`, `nmap`, `traceroute`

```
daa5fb086acb   fedora-central                          "/bin/bash"
   fedora_central
```

3.2 Ejecución en red compartida

```
docker run -dit --name fedora_central --network host fedora-central
```

3.3 Verificación de conectividad

- Ping exitoso a VMs y contenedores
- Escaneo de red con `nmap`

Cada máquina virtual conectada a `bridge-net` recibió una dirección IP válida dentro del rango definido:

- Rocky Linux: 192.168.200.188

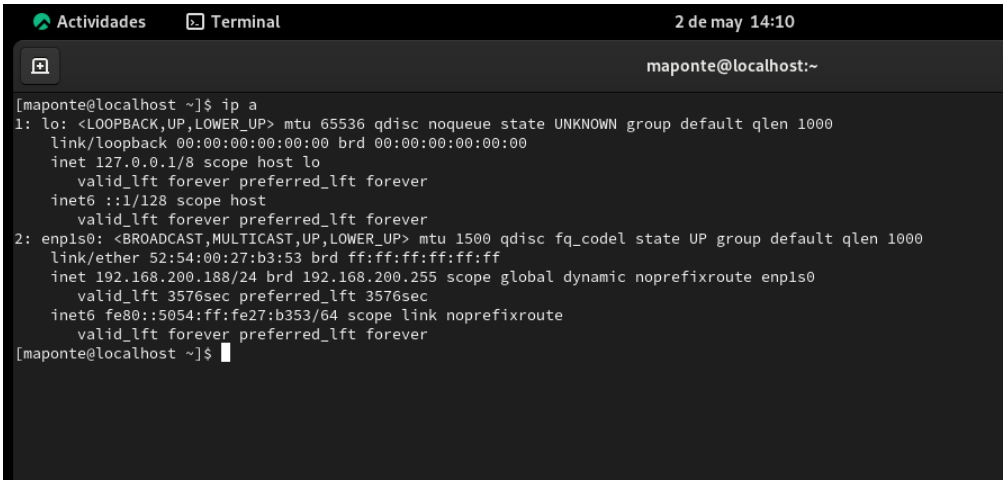


Figura 18: ip a para Rocky

- Arch Linux: 192.168.200.195

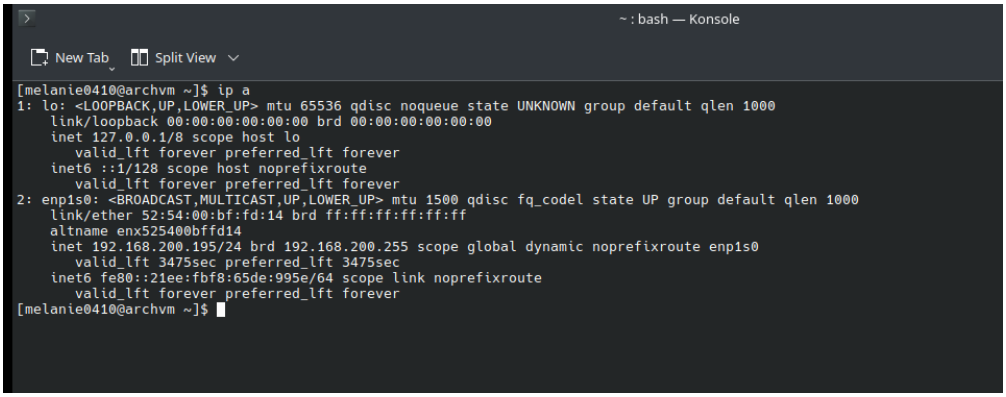


Figura 19: ip a para Arch

- Manjaro Linux: 192.168.200.234

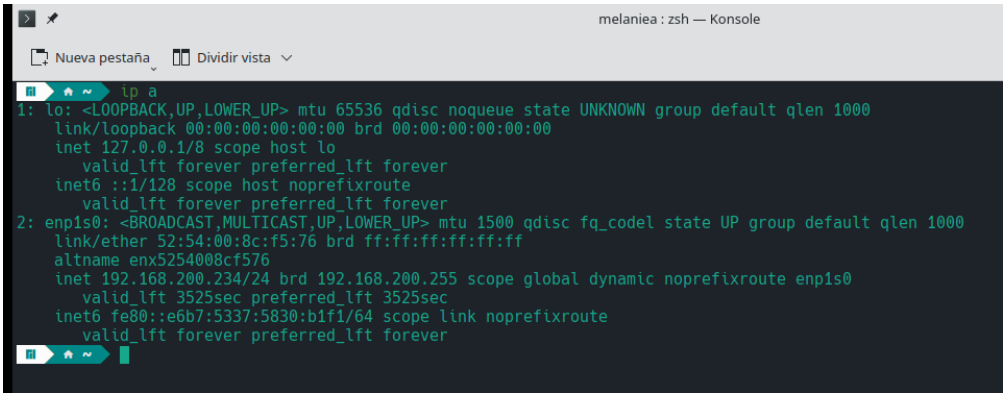
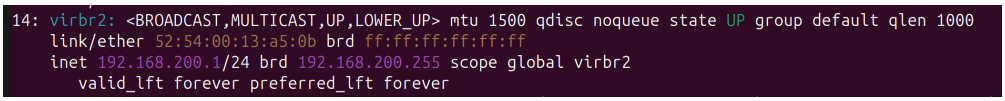


Figura 20: ip a para Manjaro

- `ip a`: para verificar la IP del contenedor. Se observó que utiliza la red `192.168.200.0/24` a través de la interfaz `virbr2`.



- `ping <ip-vm>`: se realizó ping a cada máquina virtual:
  - Rocky Linux: 192.168.200.188

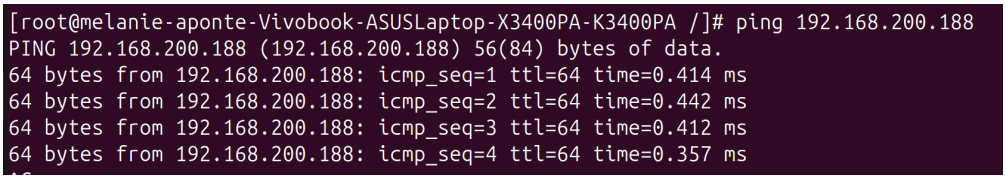


Figura 21: Ping a rocky desde el contenedor

  - Arch Linux: 192.168.200.195

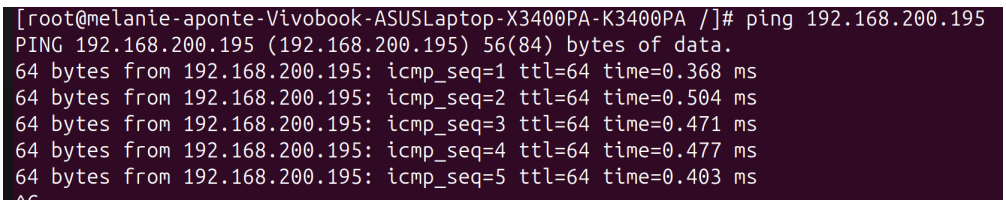


Figura 22: Ping a Arch desde el contenedor

- Manjaro Linux: 192.168.200.234

```
[root@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA /]# ping 192.168.200.234
PING 192.168.200.234 (192.168.200.234) 56(84) bytes of data.
64 bytes from 192.168.200.234: icmp_seq=1 ttl=64 time=0.449 ms
64 bytes from 192.168.200.234: icmp_seq=2 ttl=64 time=0.386 ms
64 bytes from 192.168.200.234: icmp_seq=3 ttl=64 time=0.408 ms
64 bytes from 192.168.200.234: icmp_seq=4 ttl=64 time=0.368 ms
^C
```

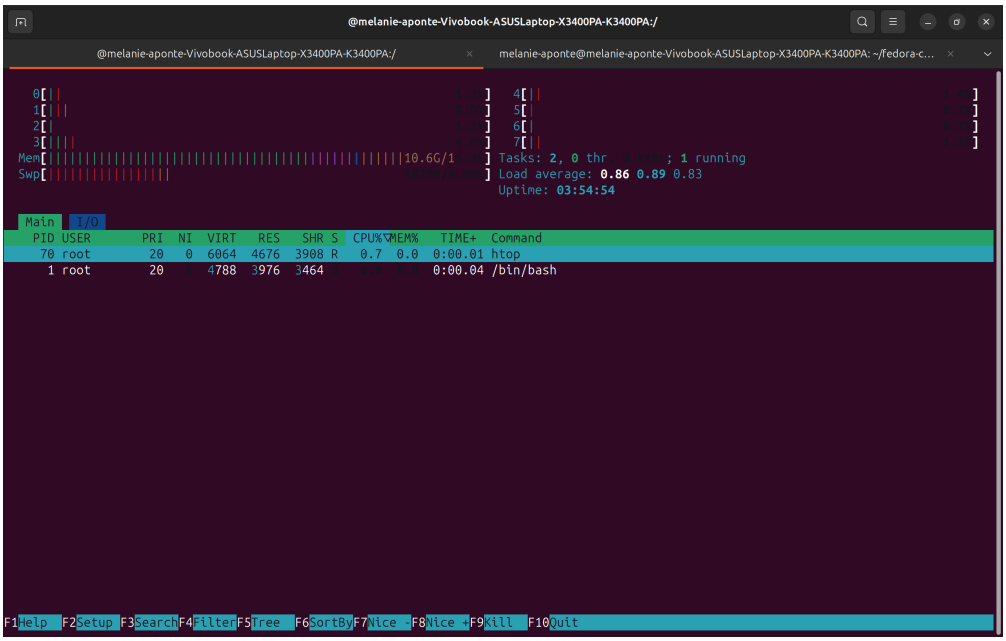
Figura 23: Ping a manjaro desde el contenedor

Todas respondieron exitosamente.

- `traceroute <ip-vm>`: para ver la ruta hacia las máquinas virtuales.

```
[root@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA /]# traceroute 192.168.200.195
traceroute to 192.168.200.195 (192.168.200.195), 30 hops max, 60 byte packets
 1  192.168.200.195 (192.168.200.195)  0.505 ms  0.467 ms  0.426 ms
[root@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA /]# traceroute 192.168.200.234
traceroute to 192.168.200.234 (192.168.200.234), 30 hops max, 60 byte packets
 1  192.168.200.234 (192.168.200.234)  0.596 ms  0.540 ms  0.516 ms
[root@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA /]# traceroute 192.168.200.188
traceroute to 192.168.200.188 (192.168.200.188), 30 hops max, 60 byte packets
 1  192.168.200.188 (192.168.200.188)  0.644 ms !X  0.588 ms !X  0.569 ms !X
[root@melanie-aponte-Vivobook-ASUSLaptop-X3400PA-K3400PA /]#
```

- `ps aux` y `htop`: para observar procesos en ejecución dentro del contenedor central.



Esto permitió comprobar que el contenedor central podía comunicarse con el resto del entorno virtualizado, cumpliendo el objetivo de ser el nodo central de análisis.

4 Análisis de Red y Procesos (Inicio)

4.1 Escaneo de puertos

```
nmap -Pn 192.168.200.130
nmap -Pn 172.25.0.2
```

4.2 Comparación de servicios y uso de recursos

- `ps aux`, `htop`, `netstat -tulpn`, `ip a`, `docker inspect`

Entorno	Interfaz	¿Tcpdump instalado?	¿Captura tráfico?
Rocky Linux (VM)	enp0s3	Sí (dnf)	Sí
Manjaro (VM)	enp0s3	Sí (pacman)	Sí
Arch Linux (VM)	enp0s3	Sí (pacman)	Sí
Debian (Contenedor)	eth0	Sí (apt)	Sí
Alpine (Contenedor)	eth0	Sí (apk)	Sí
Garuda/Arch (Contenedor)	eth0	Sí (pacman)	Sí

Cuadro 1: Comparativa del uso de `tcpdump` en máquinas virtuales y contenedores

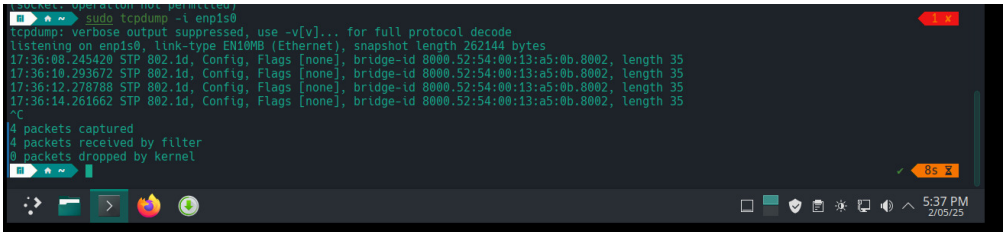


Figura 24: Tcp manjaro

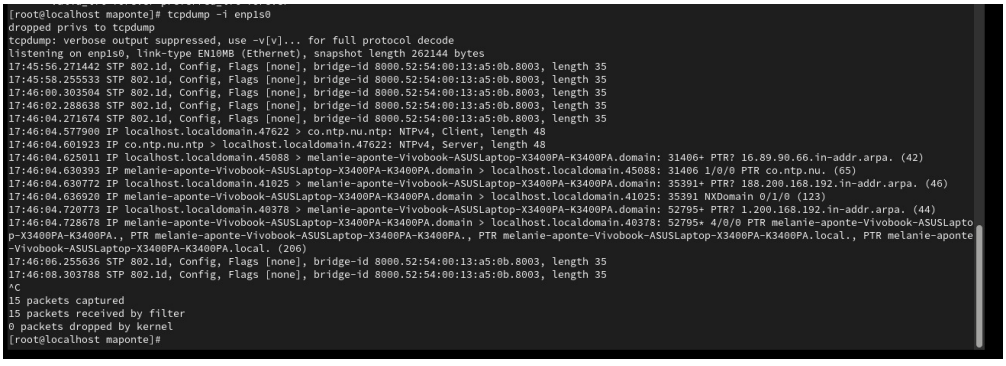


Figura 25: Tcp Rocky

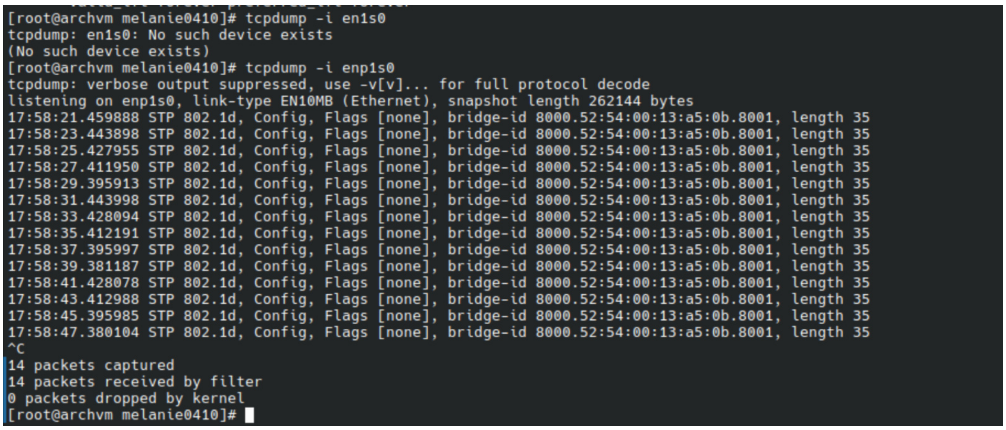


Figura 26: Tcp Arch

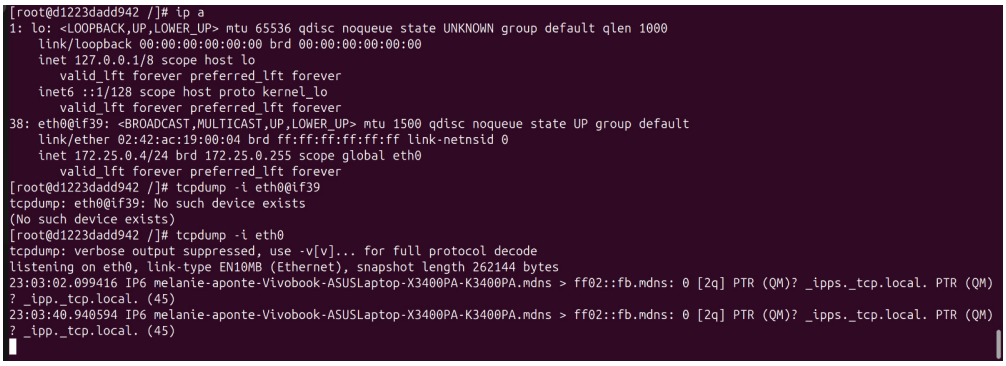


Figura 27: Tcp garuda

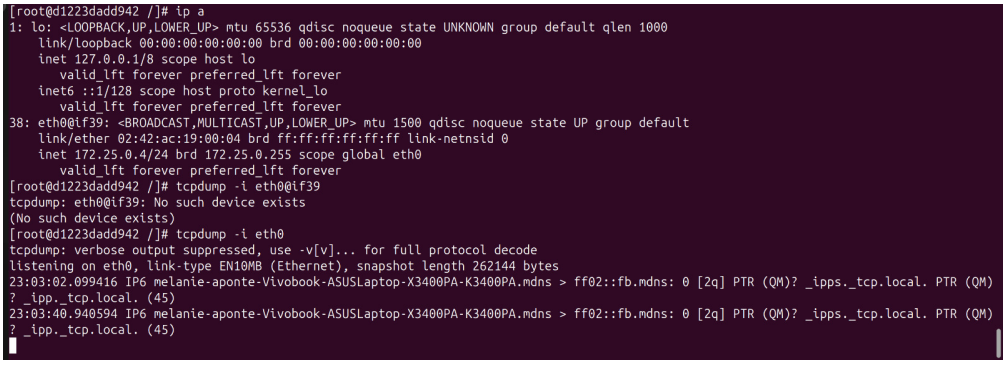


Figura 28: Tcp Debian

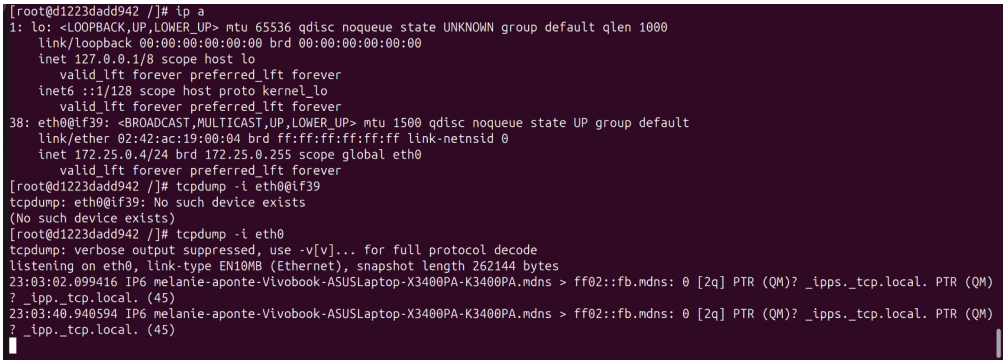


Figura 29: Tcp Alpine

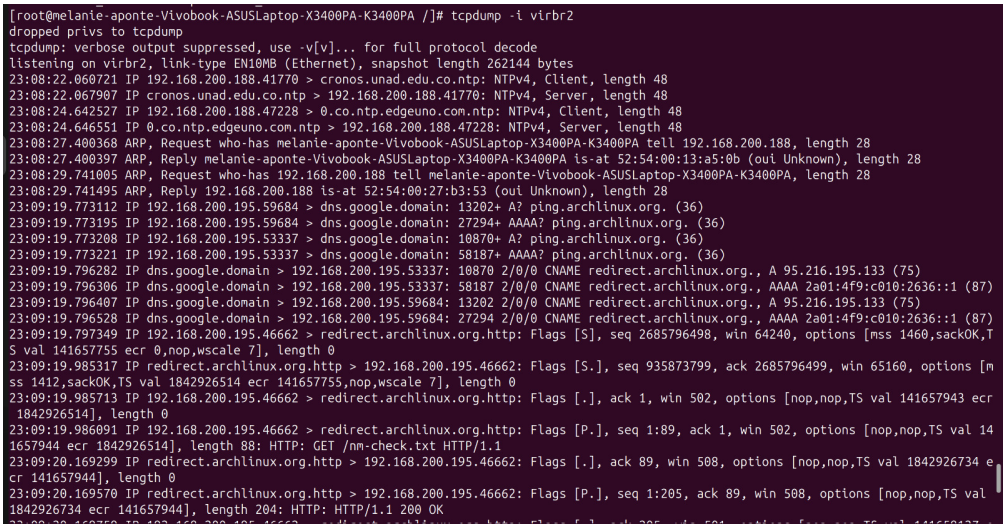


Figura 30: Tcp contendor central

5 Integración y uso de herramientas como Grafana, Prometheus y Zabbix.

5.1 Imagenes de resultados

- Estos contendeores se encuentran en una red llamada red monitorizacion

El docker-compose.yml es el siguiente:

```yaml
version: '3'

services:

  prometheus:
    image: prom/prometheus
    container_name: prometheus
    ports:
      - "9091:9090"
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    networks:
      - red_monitorizacion

  grafana:
    image: grafana/grafana
    container_name: grafana
    ports:
      - "3001:3000"
    networks:
      - red_monitorizacion

  zabbix-server:
    image: zabbix/zabbix-server-pgsql
    container_name: zabbix-server
    environment:
       DB_SERVER_HOST: zabbix-db
    ports:
      - "10051:10051"
    networks:
      - red_monitorizacion

  zabbix-db:
    image: mysql:8.0
    environment:
      MYSQL_DATABASE: zabbix
      MYSQL_USER: zabbix
      MYSQL_PASSWORD: zabbix
      MYSQL_ROOT_PASSWORD: zabbix
    networks:
      - red_monitorizacion

  zabbix-web:
    image: zabbix/zabbix-web-nginx-mysql
    container_name: zabbix-web
    environment:
      DB_SERVER_HOST: zabbix-db
      DB_SERVER_PORT: 3306
      DB_SERVER_DBNAME: zabbix
      MYSQL_USER: zabbix
      MYSQL_PASSWORD: zabbix
      MYSQL_ROOT_PASSWORD: zabbix

    ports:
      - "8081:8080"
    networks:
      - red_monitorizacion

networks:
  red_monitorizacion:
```

El prometheus.yml es el siguiente:

```yaml
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

scrape_configs:
  - job_name: 'docker'
    static_configs:
      - targets: ['zabbix-agent:10050', 'node-exporter:9100']

  - job_name: 'qemu-vms'
    static_configs:
      - targets: ['192.168.200.188:9100', '192.168.200.195:9100', '192.168.200.234:9100']
```

Usando htop en grafana, zabbix y prometheus

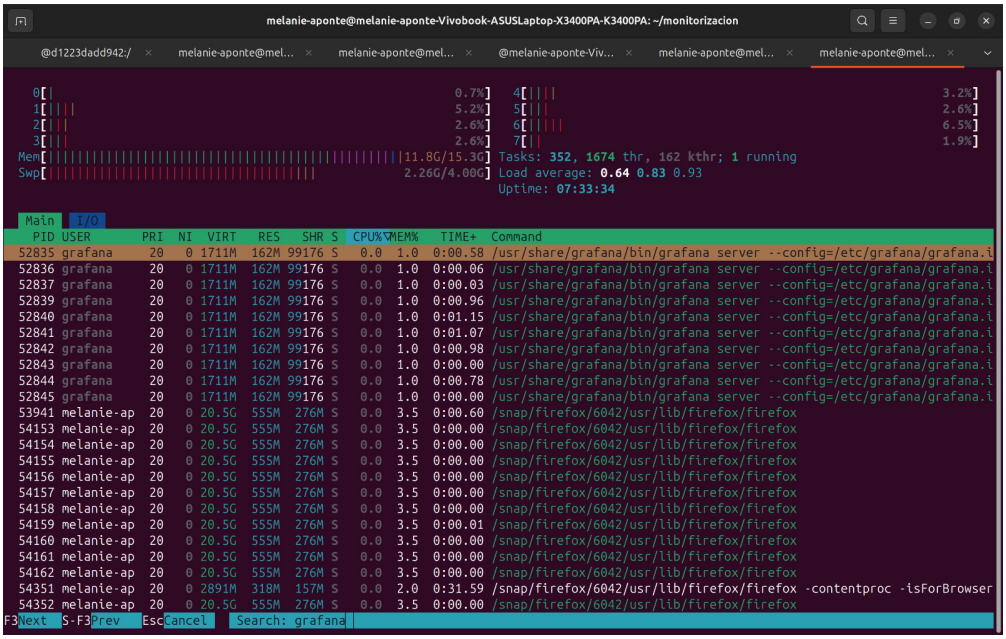


Figura 31: Htop grafana

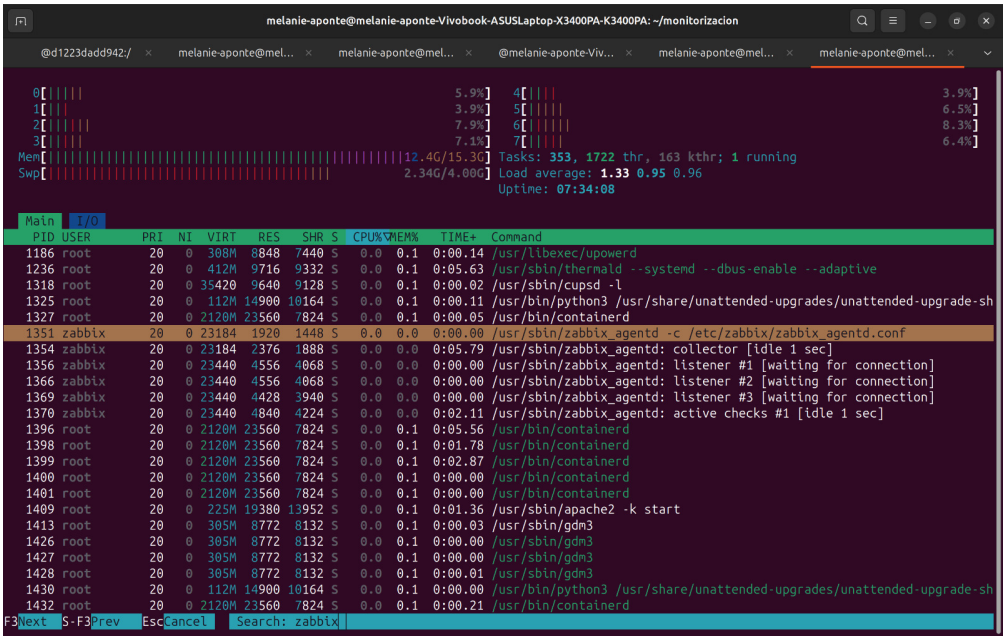


Figura 32: Htop zabbix

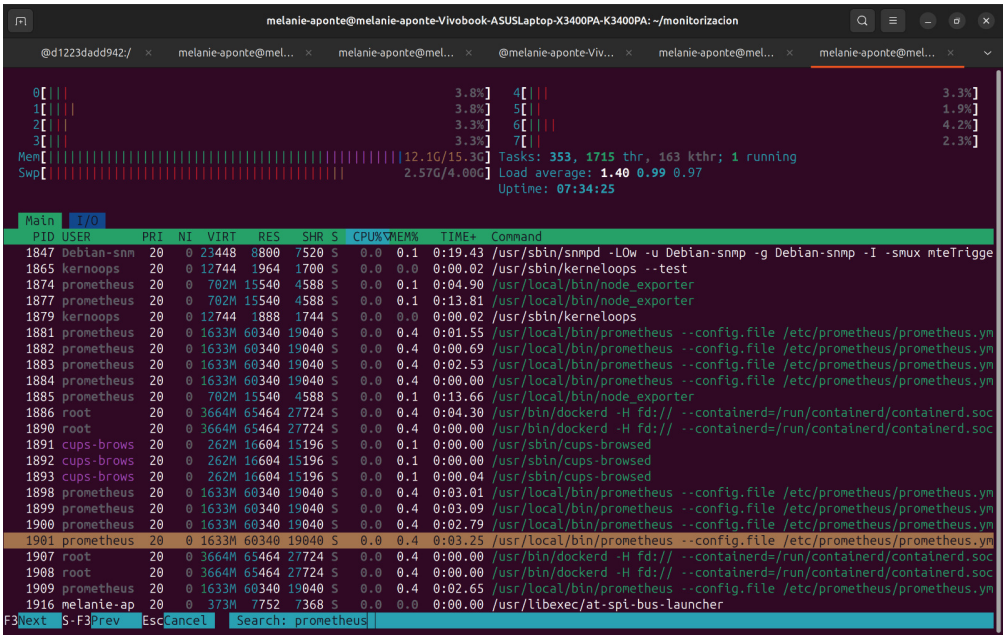
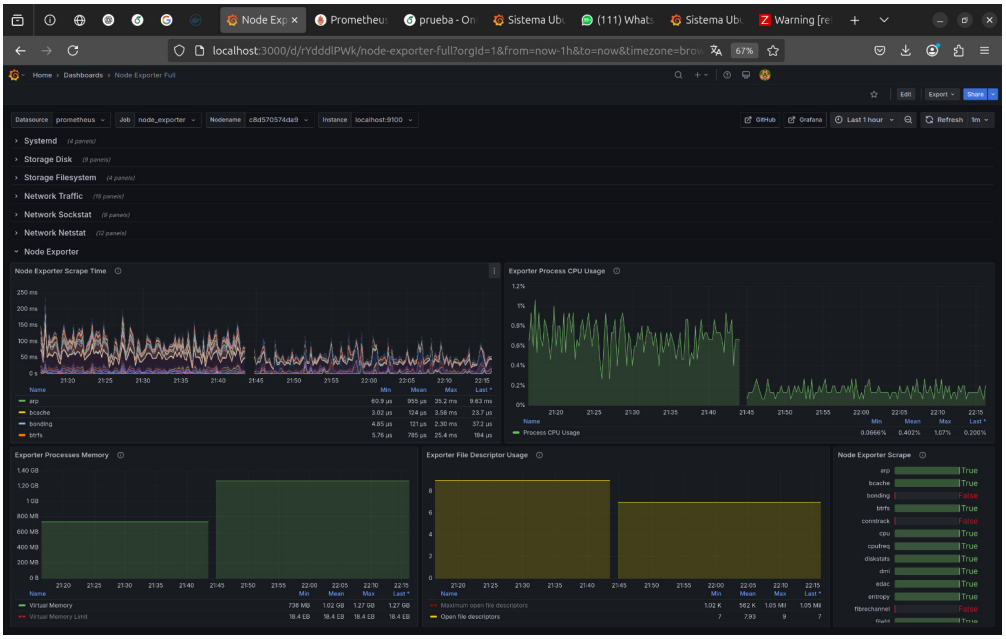


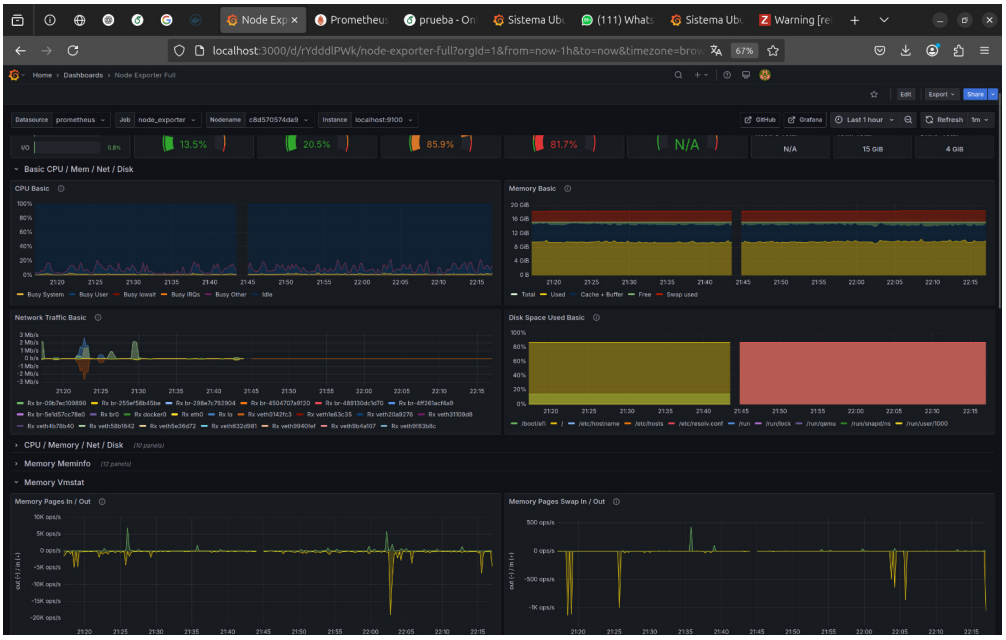
Figura 33: Htop prometheus

Para ver que los puertos están .ºescuchando”:

```
tcp6       0      0 :::3000                 :::*                    ESCUCHAR    -
tcp6       0      0 :::3001                 :::*                    ESCUCHAR    -
tcp6       0      0 :::80                   :::*                    ESCUCHAR    -
tcp6       0      0 :::9100                 :::*                    ESCUCHAR    -
tcp6       0      0 :::9090                 :::*                    ESCUCHAR    -
tcp6       0      0 :::9091                 :::*                    ESCUCHAR    -
tcp6       0      0 :::10050                :::*                    ESCUCHAR    -
tcp6       0      0 :::10051                :::*                    ESCUCHAR    -
tcp6       0      0 ::1:631                 :::*                    ESCUCHAR    -
tcp6       0      0 :::8080                 :::*                    ESCUCHAR    -
```

A grafana se accede por el enlace: http://localhost:3000





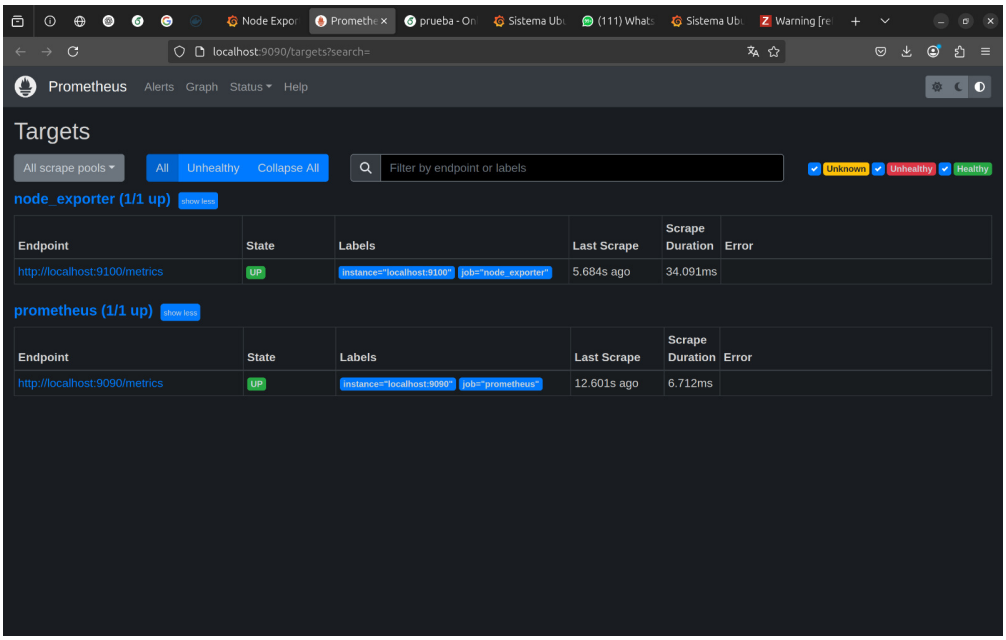


Figura 34: Prometheus

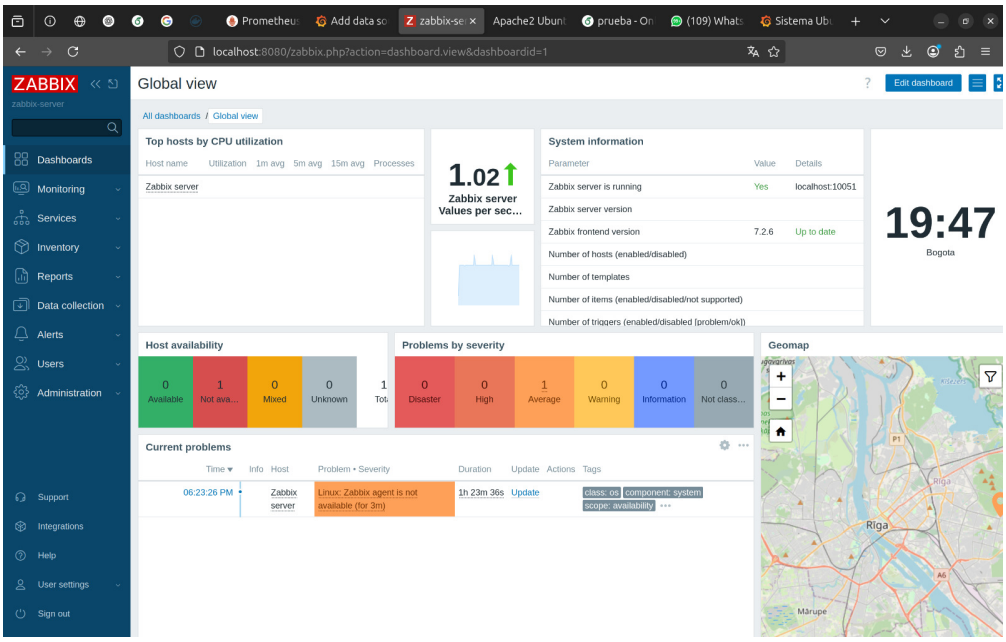


Figura 35: Zabbix