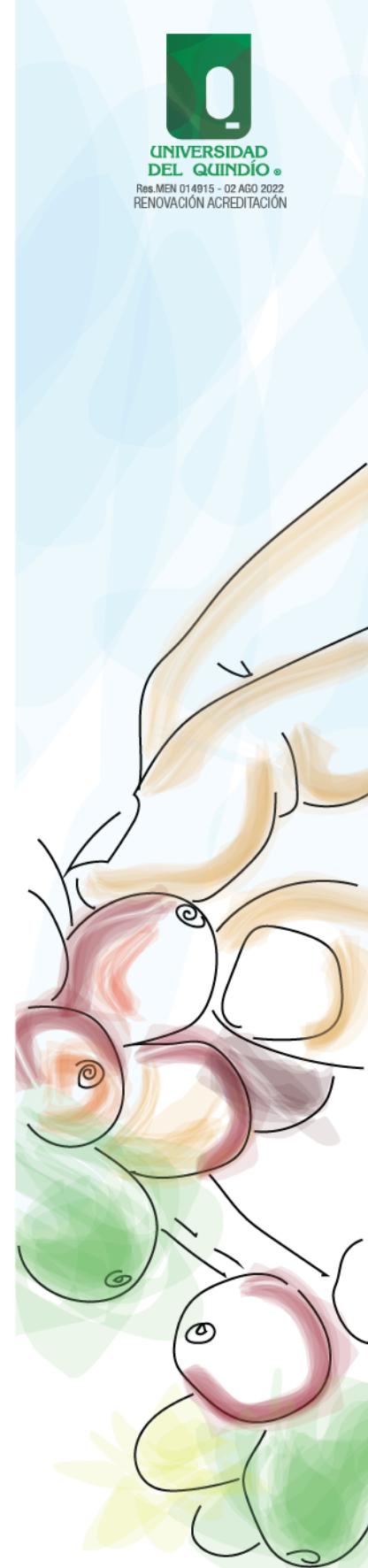


Proyecto final: Infraestructura computacional

Juan Esteban Galeano Bolaños
Mariana Pineda Muñoz
Santiago Rodas Echeverry
CC: 1005087822
CC: 1095550335
CC: 1092851226

Ingeniería de sistemas y computación
Docente: Maycol Cárdenas Acevedo

2025



Introducción

El presente proyecto tiene como propósito la implementación de una infraestructura computacional basada en tecnologías de virtualización y contenedorización, con el fin de integrar servicios distribuidos de manera eficiente y segura.

A través de la creación de arreglos RAID 1, la configuración de volúmenes lógicos LVM y la ejecución de contenedores Docker y Podman, se busca garantizar la persistencia, tolerancia a fallos y portabilidad de los servicios.

Los servicios desplegados —Apache, MySQL y Nginx— representan una arquitectura típica de servidores web y de base de datos en entornos empresariales.

El desarrollo de este proyecto permite afianzar conocimientos en almacenamiento redundante, administración de volúmenes, virtualización, gestión de contenedores y automatización de despliegues, pilares fundamentales de la infraestructura moderna en la nube.

Marco teórico

RAID (Redundant Array of Independent Disks) combina múltiples discos duros en un solo sistema lógico para mejorar el rendimiento y la tolerancia a fallos. En este proyecto se utilizó RAID 1 (espejo), que duplica los datos en dos discos para garantizar la integridad ante fallos.

LVM (Logical Volume Manager) permite administrar el almacenamiento de forma flexible mediante volúmenes físicos (PV), grupos de volúmenes (VG) y volúmenes lógicos (LV), posibilitando ampliaciones y respaldos sin interrupciones.

La virtualización crea máquinas virtuales independientes, mientras que la contenedorización, a través de Docker y Podman, aísla aplicaciones en entornos ligeros. Docker utiliza un demonio central, mientras que Podman opera sin daemon, ofreciendo mayor seguridad y compatibilidad con imágenes Docker.

Los servicios implementados fueron: Apache (servidor web), MySQL (gestor de bases de datos), Nginx (proxy inverso y balanceador) y phpMyAdmin (interfaz web para

MySQL). La persistencia se logró montando volúmenes LVM sobre los contenedores, asegurando la conservación de datos incluso tras reinicios.

Definiciones clave

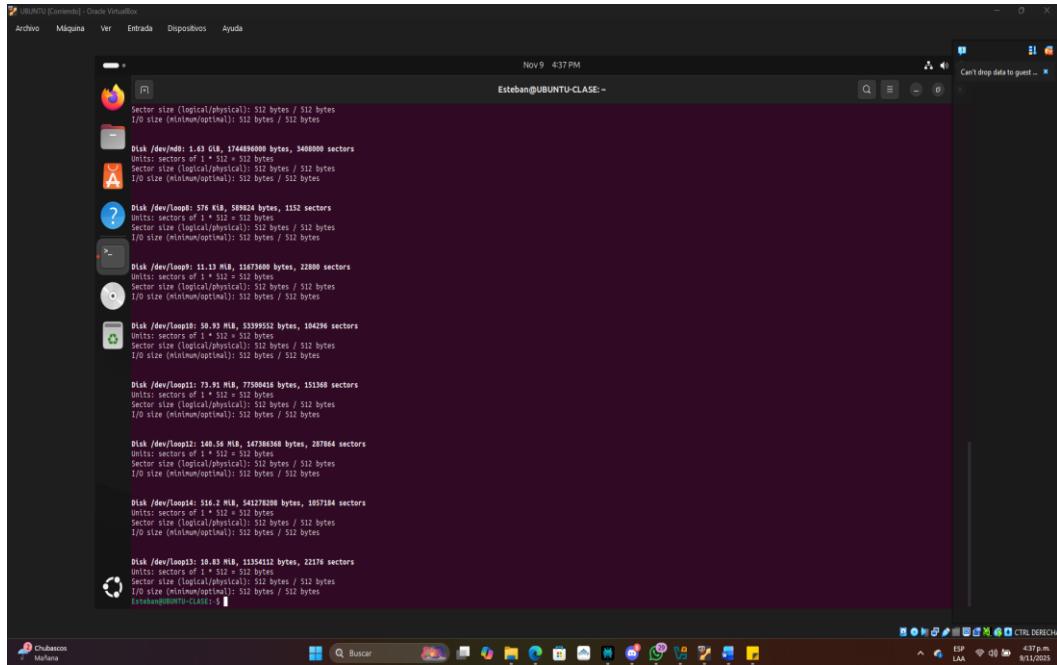
- Contenedor: Entorno aislado que ejecuta una aplicación junto con sus dependencias.
- Imagen: Plantilla inmutable que contiene el sistema base y archivos necesarios para crear un contenedor.
- Volumen: Directorio persistente montado dentro del contenedor para conservar datos.
- Dockerfile: Archivo de instrucciones para construir una imagen personalizada.
- Pod: Conjunto de contenedores que comparten red y almacenamiento, concepto adoptado por Podman y Kubernetes.

Actividades

Estructura del proyecto:

```
Esteban@UBUNTU-CLASE:~$ tree ~/docker_builds && sudo tree -L 2 /mnt
~/docker_builds
├── apache
│   └── Dockerfile
├── mysql
│   ├── Dockerfile
│   └── nginx
│       ├── Dockerfile
│       └── index.html
4 directories, 5 files
/mnt
└── apache_vol
    └── lost+found
    └── backup
        └── CarpetaCompartida
            └── mysql
                ├── auto.cnf
                ├── binlog.000001
                ├── binlog.000002
                ├── binlog.index
                ├── ca-key.pem
                ├── ca.pem
                ├── client-cert.pem
                ├── clientes
                │   └── client-key.pem
                ├── #ib_16384_0 dblwr
                ├── #ib_16384_1 dblwr
                └── ibuffer_pool
                    └── ibdata1
                        └── ibtmp1
                    └── #innodb redo
                    └── #innodb temp
                    └── lost+found
                └── mysql
                    ├── mysql.ibd
                    └── mysql.sock -> /var/run/mysqld/mysqld.sock
                └── performance_schema
                    ├── private_key.pem
                    ├── public_key.pem
                    ├── server-cert.pem
                    └── server-key.pem
                └── sys
                    ├── undo_001
                    └── undo_002
            └── nginx_vol
                └── lost+found
15 directories, 21 files
Esteban@UBUNTU-CLASE:~$
```

1. Verificar los discos disponibles sudo fdisk -l



Observación: Se listan los discos conectados al sistema y se identifican los discos para empezar con la fase 1

Propósito	Disco 1	Disco 2	Resultado
Apache	/dev/sdb → APACHE.vdi	/dev/sdc → PRUEBA1.vdi	/dev/md0
MySQL	/dev/sdd → MySQL.vdi	/dev/sde → PRUEBA2.vdi	/dev/md1
Nginx	/dev/sdf → Nginx1vdi.vdi	/dev/sdg → PRUEBA3.vdi	/dev/md2



Almacenamiento	
Controlador:	IDE
Dispositivo IDE primario 0:	[Unidad óptica] VBoxGuestAdditions.iso (50,69 MB)
Controlador:	SATA
Puerto SATA 0:	UBUNTU.vdi (Normal, 25,00 GB)
Puerto SATA 1:	PRUEBA1.vdi (Normal, 1,63 GB)
Puerto SATA 2:	PRUEBA2vdi.vdi (Normal, 2,28 GB)
Puerto SATA 3:	PRUEBA3.vdi (Normal, 2,28 GB)
Puerto SATA 4:	APACHE.vdi (Normal, 1,71 GB)
Puerto SATA 5:	MySQL.vdi (Normal, 1,41 GB)
Puerto SATA 6:	Nginx1vdi.vdi (Normal, 2,01 GB)

- **FASE 1 — Configuración de RAID 1:** El objetivo será crear 3 arreglos RAID 1 (espejo) con los discos virtuales disponibles. Además, cada RAID servirá luego como base para un volumen LVM que usará un contenedor diferente.

```
Esteban@UBUNTU-CLASE:~$ sudo mdadm --create --verbose /dev/md0 --level=1 --raid-devices=2 /dev/sdb /dev/sdc
sudo mdadm --create --verbose /dev/md1 --level=1 --raid-devices=2 /dev/sdd /dev/sde
sudo mdadm --create --verbose /dev/md2 --level=1 --raid-devices=2 /dev/sdf /dev/sdg
mdadm: Note: this array has metadata at the start and
      may not be suitable as a boot device. If you plan to
      store '/boot' on this device please ensure that
      your boot-loader understands md/v1.x metadata, or use
      --metadata=0.90
mdadm: size set to 1704000K
mdadm: largest drive (/dev/sdc) exceeds size (1704000K) by more than 1%
Continue creating array?
Continue creating array? (y/n) y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
mdadm: Note: this array has metadata at the start and
      may not be suitable as a boot device. If you plan to
      store '/boot' on this device please ensure that
      your boot-loader understands md/v1.x metadata, or use
      --metadata=0.90
mdadm: size set to 1788224K
mdadm: largest drive (/dev/sdd) exceeds size (1788224K) by more than 1%
Continue creating array?
Continue creating array? (y/n) y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md1 started.
mdadm: Note: this array has metadata at the start and
      may not be suitable as a boot device. If you plan to
      store '/boot' on this device please ensure that
      your boot-loader understands md/v1.x metadata, or use
      --metadata=0.90
mdadm: size set to 1477952K
mdadm: largest drive (/dev/sdg) exceeds size (1477952K) by more than 1%
Continue creating array?
Continue creating array? (y/n) y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md2 started.
Esteban@UBUNTU-CLASE:~$
```

Observación: Los códigos que se emplearon fueron los siguientes para la creación de los respectivos RAIDS:

RAID 1 para Apache

```
sudo mdadm --create --verbose /dev/md0 --level=1 --raid-devices=2 /dev/sdb
/dev/sdc
```

RAID 1 para MySQL

```
sudo mdadm --create --verbose /dev/md1 --level=1 --raid-devices=2 /dev/sdd
/dev/sde
```

RAID 1 para Nginx

```
sudo mdadm --create --verbose /dev/md2 --level=1 --raid-devices=2 /dev/sdf
/dev/sdg
```

- **Explicación del funcionamiento de los comandos:**

- `--create` → crea un nuevo arreglo RAID.
- `--level=1` → indica RAID 1 (espejo).
- `--raid-devices=2` → usa dos discos por arreglo.
- Cada arreglo genera un nuevo dispositivo virtual `/dev/mdX`.

2. Verificación del estado de los RAID y guardar su configuración:

```
Esteban@UBUNTU-CLASE:~$ cat /proc/mdstat
Personalities : [raid1] [linear] [raid0] [raid6] [raid5] [raid4] [raid10]
md2 : active raid1 sdg[1] sdf[0]
      1477952 blocks super 1.2 [2/2] [UU]

md1 : active raid1 sde[1] sdd[0]
      1788224 blocks super 1.2 [2/2] [UU]

md0 : active raid1 sdc[1] sdb[0]
      1704000 blocks super 1.2 [2/2] [UU]

unused devices: <none>
Esteban@UBUNTU-CLASE:~$
```

Observación: Se verifica el estado actual de los arreglos RAID. Esta acción permite registrar la configuración de los RAIDs en el sistema, garantizando su montaje automático al reiniciar el equipo. Los comandos ejecutados fueron los siguientes:

- ❖ `cat /proc/mdstat`: Verificación del estado de los arreglos RAID activos.



❖ sudo mdadm --detail --scan | sudo tee -a /etc/mdadm/mdadm.conf:

Registro de la información de los RAIDs en el archivo de configuración del sistema.

❖ sudo update-initramfs -u: Actualización del sistema para guardar y aplicar la configuración de los RAIDs.

```
Esteban@UBUNTU-CLASE:~$ sudo mdadm --detail --scan | sudo tee -a /etc/mdadm/mdadm.conf
sudo update-initramfs -u
ARRAY /dev/md0 metadata=1.2 UUID=c9f04330:9a02d3d1:ca409e76:238d8770
ARRAY /dev/md1 metadata=1.2 UUID=856220f2:ae23b86e:3a45f0d6:1e785408
ARRAY /dev/md2 metadata=1.2 UUID=2b3fe041:212f6a9f:e435a7b8:f00241a4
update-initramfs: Generating /boot/initrd.img-6.14.0-35-generic
Esteban@UBUNTU-CLASE:~$
```

- Verificar cada RAID en detalle

```
Esteban@UBUNTU-CLASE:~$ sudo mdadm --detail /dev/md0
sudo mdadm --detail /dev/md1
sudo mdadm --detail /dev/md2
/dev/md0:
          Version : 1.2
          Creation Time : Sun Nov  9 16:48:41 2025
          Raid Level : raid1
          Array Size : 1704000  (1664.06 MiB 1744.90 MB)
          Used Dev Size : 1704000  (1664.06 MiB 1744.90 MB)
          Raid Devices : 2
          Total Devices : 2
          Persistence : Superblock is persistent

          Update Time : Sun Nov  9 16:48:50 2025
          State : clean
          Active Devices : 2
          Working Devices : 2
          Failed Devices : 0
          Spare Devices : 0

Consistency Policy : resync

              Name : UBUNTU-CLASE:0  (local to host UBUNTU-CLASE)
              UUID : c9f04330:9a02d3d1:ca409e76:238d8770
              Events : 17

          Number  Major  Minor  RaidDevice State
              0      8        16          0      active sync   /dev/sdb
              1      8        32          1      active sync   /dev/sdc
/dev/md1:
```



```
/dev/md1:
      Version : 1.2
      Creation Time : Sun Nov  9 16:48:44 2025
      Raid Level : raid1
      Array Size : 1788224 (1746.31 MiB 1831.14 MB)
      Used Dev Size : 1788224 (1746.31 MiB 1831.14 MB)
      Raid Devices : 2
      Total Devices : 2
      Persistence : Superblock is persistent

      Update Time : Sun Nov  9 16:48:53 2025
                     State : clean
      Active Devices : 2
      Working Devices : 2
      Failed Devices : 0
      Spare Devices : 0

      Consistency Policy : resync

              Name : UBUNTU-CLASE:1 (local to host UBUNTU-CLASE)
              UUID : 856220f2:ae23b86e:3a45f0d6:1e785408
              Events : 17

      Number  Major  Minor  RaidDevice State
          0      8      48        0  active sync  /dev/sdd
          1      8      64        1  active sync  /dev/sde
```

```
/dev/md2:
      Version : 1.2
      Creation Time : Sun Nov  9 16:48:48 2025
      Raid Level : raid1
      Array Size : 1477952 (1443.31 MiB 1513.42 MB)
      Used Dev Size : 1477952 (1443.31 MiB 1513.42 MB)
      Raid Devices : 2
      Total Devices : 2
      Persistence : Superblock is persistent

      Update Time : Sun Nov  9 16:48:55 2025
                     State : clean
      Active Devices : 2
      Working Devices : 2
      Failed Devices : 0
      Spare Devices : 0

      Consistency Policy : resync

              Name : UBUNTU-CLASE:2 (local to host UBUNTU-CLASE)
              UUID : 2b3fe041:212f6a9f:e435a7b8:f00241a4
              Events : 17

      Number  Major  Minor  RaidDevice State
          0      8      80        0  active sync  /dev/sdf
          1      8      96        1  active sync  /dev/sdg
Esteban@UBUNTU-CLASE:~$ █
```

Posteriormente, se consultan los detalles de cada arreglo RAID mediante los siguientes comandos:

- sudo mdadm --detail /dev/md0
- sudo mdadm --detail /dev/md1
- sudo mdadm --detail /dev/md2

Al ejecutar estos comandos, se muestra información detallada sobre cada dispositivo, incluyendo:

- Estado: active
- Nivel: raid1
- Discos activos y sincronizados
- Tamaño total disponible

Esta verificación permite confirmar que los arreglos RAID se encuentran operativos y correctamente sincronizados, asegurando la integridad de los datos y la disponibilidad del sistema.

- **FASE 2 — Configuración de LVM sobre RAID**

Objetivo: Crear la capa de administración de volúmenes lógicos sobre los arreglos RAID existentes, siguiendo tres metas:

1. Crear un Physical Volume (PV) sobre cada dispositivo RAID administrado con mdadm.

```
Esteban@UBUNTU-CLASE:~$ sudo pvcreate /dev/md0
sudo pvcreate /dev/md1
sudo pvcreate /dev/md2
[sudo] password for Esteban:
WARNING: ext4 signature detected on /dev/md0 at offset 1080. Wipe it? [y/n]: Y
Wiping ext4 signature on /dev/md0.
Physical volume "/dev/md0" successfully created.
Physical volume "/dev/md1" successfully created.
Physical volume "/dev/md2" successfully created.
Esteban@UBUNTU-CLASE:~$
```

- Se preparan los arreglos RAID para su gestión con LVM, convirtiendo cada uno en un volumen físico reconocible por el sistema. Para ello, se inicializa LVM sobre cada dispositivo RAID con los siguientes comandos:

- ❖ sudo pvcreate /dev/md0
- ❖ sudo pvcreate /dev/md1
- ❖ sudo pvcreate /dev/md2



- Con esta acción, cada RAID queda listo para integrarse en grupos de volúmenes y permitir la creación de volúmenes lógicos destinados a los datos de los servicios.
2. Definir un Volume Group (VG) y crear grupos de almacenamiento independientes para cada servicio. Cada servicio debe contar con su propio espacio de almacenamiento, de manera que no interfiera con los demás. Esta separación permite organizar, expandir y mantener el almacenamiento de manera clara y ordenada, asegurando que cada servicio se gestione por separado y de forma eficiente.

```
Esteban@UBUNTU-CLASE:~$ sudo vgcreate vg_apache /dev/md0
sudo vgcreate vg_mysql /dev/md1
sudo vgcreate vg_nginx /dev/md2
  Volume group "vg_apache" successfully created
  Volume group "vg_mysql" successfully created
  Volume group "vg_nginx" successfully created
Esteban@UBUNTU-CLASE:~$
```

- ❖ sudo vgcreate vg_apache /dev/md0
- ❖ sudo vgcreate vg_mysql /dev/md1
- ❖ sudo vgcreate vg_nginx /dev/md2

3. Provisionar un Logical Volume (LV) por servicio para alojar los datos

de cada contenedor.

```
Esteban@UBUNTU-CLASE:~$ sudo lvcreate -l 100%FREE -n lv_apache vg_apache
sudo lvcreate -l 100%FREE -n lv_mysql vg_mysql
sudo lvcreate -l 100%FREE -n lv_nginx vg_nginx
[sudo] password for Esteban:
  Logical volume "lv_apache" created.
  Logical volume "lv_mysql" created.
  Logical volume "lv_nginx" created.
Esteban@UBUNTU-CLASE:~$
```

- ❖ sudo lvcreate -l 100%FREE -n lv_apache vg_apache
- ❖ sudo lvcreate -l 100%FREE -n lv_mysql vg_mysql
- ❖ sudo lvcreate -l 100%FREE -n lv_nginx vg_nginx

La opción -l 100%FREE crea el volumen usando todo el espacio libre disponible del grupo de almacenamiento. Cada volumen lógico generará un dispositivo accesible en el sistema:

- ❖ /dev/vg_apache/lv_apache
- ❖ /dev/vg_mysql/lv_mysql
- ❖ /dev/vg_nginx/lv_nginx

4. Formatear los volúmenes con EXT4 para dejarlos listos para su uso por los servicios. Se aplica el sistema de archivos ext4 a cada volumen lógico con los siguientes comandos:

- ❖ sudo mkfs.ext4 /dev/vg_apache/lv_apache
- ❖ sudo mkfs.ext4 /dev/vg_mysql/lv_mysql
- ❖ sudo mkfs.ext4 /dev/vg_nginx/lv_nginx

Con este paso, cada volumen queda preparado para ser montado y utilizado por los contenedores correspondientes.



```
Esteban@UBUNTU-CLASE:~$ sudo mkfs.ext4 /dev/vg_apache/lv_apache
sudo mkfs.ext4 /dev/vg_mysql/lv_mysql
sudo mkfs.ext4 /dev/vg_nginx/lv_nginx
[sudo] password for Esteban:
mke2fs 1.47.0 (5-Feb-2023)
Creating filesystem with 424960 4k blocks and 106288 inodes
Filesystem UUID: 3c966b02-dde5-4feb-8ef7-d30b0ffd6d65
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

mke2fs 1.47.0 (5-Feb-2023)
Creating filesystem with 446464 4k blocks and 111776 inodes
Filesystem UUID: b4eb60de-ac8f-495e-8e9e-44751889c0c6
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

mke2fs 1.47.0 (5-Feb-2023)
Creating filesystem with 368640 4k blocks and 92160 inodes
Filesystem UUID: acaa390-e039-45a7-98bf-f5f6f13e1186
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

Esteban@UBUNTU-CLASE:~$
```

5. Crear puntos de montaje

```
Esteban@UBUNTU-CLASE:~$ sudo mkdir -p /mnt/apache_vol
sudo mkdir -p /mnt/mysql_vol
sudo mkdir -p /mnt/nginx_vol
[sudo] password for Esteban:
Esteban@UBUNTU-CLASE:~$
```

Crear los directorios donde se montarán los volúmenes. Se definen los puntos de montaje para cada servicio con los siguientes comandos:

- ❖ sudo mkdir -p /mnt/apache_vol
- ❖ sudo mkdir -p /mnt/mysql_vol
- ❖ sudo mkdir -p /mnt/nginx_vol



Con esto, cada servicio dispone de una carpeta específica donde se conectará su volumen correspondiente.

6. Montar los volúmenes

```
Esteban@UBUNTU-CLASE:~$ sudo mount /dev/vg_apache/lv_apache /mnt/apache_vol
sudo mount /dev/vg_mysql/lv_mysql /mnt/mysql_vol
sudo mount /dev/vg_nginx/lv_nginx /mnt/nginx_vol
Esteban@UBUNTU-CLASE:~$
```

Montar los volúmenes para que queden disponibles en sus carpetas de trabajo. Se conectan cada volumen lógico con su punto de montaje correspondiente mediante:

- ❖ sudo mount /dev/vg_apache/lv_apache /mnt/apache_vol
- ❖ sudo mount /dev/vg_mysql/lv_mysql /mnt/mysql_vol
- ❖ sudo mount /dev/vg_nginx/lv_nginx /mnt/nginx_vol

Con este paso, los servicios pueden leer y escribir en sus ubicaciones asignadas de manera inmediata.

7. Verificar que todo quedó correctamente montado y visible en el sistema. Utiliza los siguientes comandos de comprobación:

- ❖ lsblk para visualizar los dispositivos, sus volúmenes lógicos y los puntos de montaje; confirma que /dev/vg_apache/lv_apache aparece en /mnt/apache_vol, /dev/vg_mysql/lv_mysql en /mnt/mysql_vol y /dev/vg_nginx/lv_nginx en /mnt/nginx_vol.
- ❖ mount | grep /mnt para listar únicamente los montajes bajo /mnt y verificar que cada volumen está activo en su carpeta.
- ❖ df -h para revisar capacidad total, usada y disponible de cada punto de montaje y confirmar que se reconocen como ext4.



```
Esteban@UBUNTU-CLASE:~$ lsblk
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0      7:0    0 249.2M  1 loop /snap/firefox/7084
loop1      7:1    0 249.1M  1 loop /snap/firefox/7177
loop2      7:2    0  73.9M  1 loop /snap/core22/2133
loop3      7:3    0  91.7M  1 loop /snap/gtk-common-themes/1535
loop4      7:4    0   516M  1 loop /snap/gnome-42-2204/202
loop5      7:5    0   50.8M  1 loop /snap/snapd/25202
loop6      7:6    0     4K  1 loop /snap/bare/5
loop7      7:7    0  18.5M  1 loop /snap/firmware-updater/210
loop8      7:8    0   576K  1 loop /snap/snapd-desktop-integration/315
loop9      7:9    0  11.1M  1 loop /snap/firmware-updater/167
loop10     7:10   0  50.9M  1 loop /snap/snapd/25577
loop11     7:11   0  73.9M  1 loop /snap/core22/2139
loop12     7:12   0 140.6M  1 loop /snap/docker/3265
loop13     7:13   0  10.8M  1 loop /snap/snap-store/1270
loop14     7:14   0 516.2M  1 loop /snap/gnome-42-2204/226
sda        8:0    0   25G  0 disk
└─sda1     8:1    0     1M  0 part
  └─sda2     8:2    0   25G  0 part /
sdb        8:16   0   1.6G  0 disk
└─md0      9:0    0   1.6G  0 raid1
  └─vg_apache-lv_apache 252:0  0   1.6G  0 lvm  /mnt/apache_vol
sdc        8:32   0   2.3G  0 disk
└─md0      9:0    0   1.6G  0 raid1
  └─vg_apache-lv_apache 252:0  0   1.6G  0 lvm  /mnt/apache_vol
sdd        8:48   0   2.3G  0 disk
└─md1      9:1    0   1.7G  0 raid1
  └─vg_mysql-lv_mysql  252:1  0   1.7G  0 lvm  /mnt/mysql_vol
sde        8:64   0   1.7G  0 disk
└─md1      9:1    0   1.7G  0 raid1
  └─vg_mysql-lv_mysql  252:1  0   1.7G  0 lvm  /mnt/mysql_vol
sdf        8:80   0   1.4G  0 disk
└─md2      9:2    0   1.4G  0 raid1
  └─vg_nginx-lv_nginx  252:2  0   1.4G  0 lvm  /mnt/nginx_vol
sdg        8:96   0   2G   0 disk
└─md2      9:2    0   1.4G  0 raid1
  └─vg_nginx-lv_nginx  252:2  0   1.4G  0 lvm  /mnt/nginx_vol
sr0       11:0   1  50.7M  0 rom   /media/Esteban/VBox_GAs_7.2.41
Esteban@UBUNTU-CLASE:~$
```

Servicio	RAID	VG	LV	Punto de montaje
Apache	/dev/md0	vg_apache	lv_apache	/mnt/apache_vol
MySQL	/dev/md1	vg_mysql	lv_mysql	/mnt/mysql_vol
Nginx	/dev/md2	vg_nginx	lv_nginx	/mnt/nginx_vol

- **FASE 3 — Creación de los Contenedores con Docker**

Objetivo: Crear 3 contenedores, cada uno con su servicio

Servicio	Imagen	Volumen persistente
Apache	apache_custom	/mnt/apache_vol
MySQL	mysql_custom	/mnt/mysql_vol
Nginx	nginx_custom	/mnt/nginx_vol

Cada uno usará una imagen personalizada (con su propio Dockerfile).

1. Verifica que Docker está funcionando

```
Esteban@UBUNTU-CLASE:~$ sudo systemctl status docker
[sudo] password for Esteban:
● docker.service - Docker Application Container Engine
  Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
  Active: active (running) since Sun 2025-11-09 16:27:59 -05; 6h ago
    TriggeredBy: ● docker.socket
      Docs: https://docs.docker.com
    Main PID: 1340 (dockerd)
      Tasks: 9
        Memory: 99.5M (peak: 101.0M)
          CPU: 4.896s
        CGroup: /system.slice/docker.service
                  └─1340 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Nov 09 16:27:50 UBUNTU-CLASE dockerd[1340]: time="2025-11-09T16:27:50.549165557-05:00" level=info msg="Creating a containerd client" address=/run/containerd/containerd.sock timeout=1m0s
Nov 09 16:27:51 UBUNTU-CLASE dockerd[1340]: time="2025-11-09T16:27:51.109413756-05:00" level=info msg="[graphdriver] using prior storage driver: overlay2"
Nov 09 16:27:51 UBUNTU-CLASE dockerd[1340]: time="2025-11-09T16:27:51.183455945-05:00" level=info msg="Loading containers: start."
Nov 09 16:27:58 UBUNTU-CLASE dockerd[1340]: time="2025-11-09T16:27:58.919439336-05:00" level=info msg="Loading containers: done."
Nov 09 16:27:59 UBUNTU-CLASE dockerd[1340]: time="2025-11-09T16:27:59.208426359-05:00" level=info msg="Docker daemon" commit=b9c5e0f containerd-snapshotter=false storage-driver=overlay2
Nov 09 16:27:59 UBUNTU-CLASE dockerd[1340]: time="2025-11-09T16:27:59.209850107-05:00" level=info msg="Initializing buildkit"
Nov 09 16:27:59 UBUNTU-CLASE dockerd[1340]: time="2025-11-09T16:27:59.304889694-05:00" level=info msg="Completed buildkit initialization"
Nov 09 16:27:59 UBUNTU-CLASE dockerd[1340]: time="2025-11-09T16:27:59.351841411-05:00" level=info msg="Daemon has completed initialization"
Nov 09 16:27:59 UBUNTU-CLASE dockerd[1340]: time="2025-11-09T16:27:59.362668719-05:00" level=info msg="API listen on /run/docker.sock"
Nov 09 16:27:59 UBUNTU-CLASE systemd[1]: Started docker.service - Docker Application Container Engine.
[lines 1-22/22 (END)]
```

Verificación del servicio de Docker. Se comprueba el estado del servicio para asegurar su disponibilidad operativa en el sistema. En caso de no estar activo, se procede a su activación inmediata y a su habilitación para el arranque automático en reinicios posteriores.

❖ Comprobar estado:

sudo systemctl status docker

❖ Iniciar y habilitar si no está activo:

sudo systemctl start docker

sudo systemctl enable docker

Con estas acciones, Docker queda en ejecución y configurado para iniciarse automáticamente, garantizando su disponibilidad para la gestión de contenedores.

2. Crea una carpeta para los Dockerfiles

```
Esteban@UBUNTU-CLASE:~$ mkdir -p ~/docker_builds/{apache,mysql,nginx}
cd ~/docker_builds
Esteban@UBUNTU-CLASE:~/docker_builds$
```

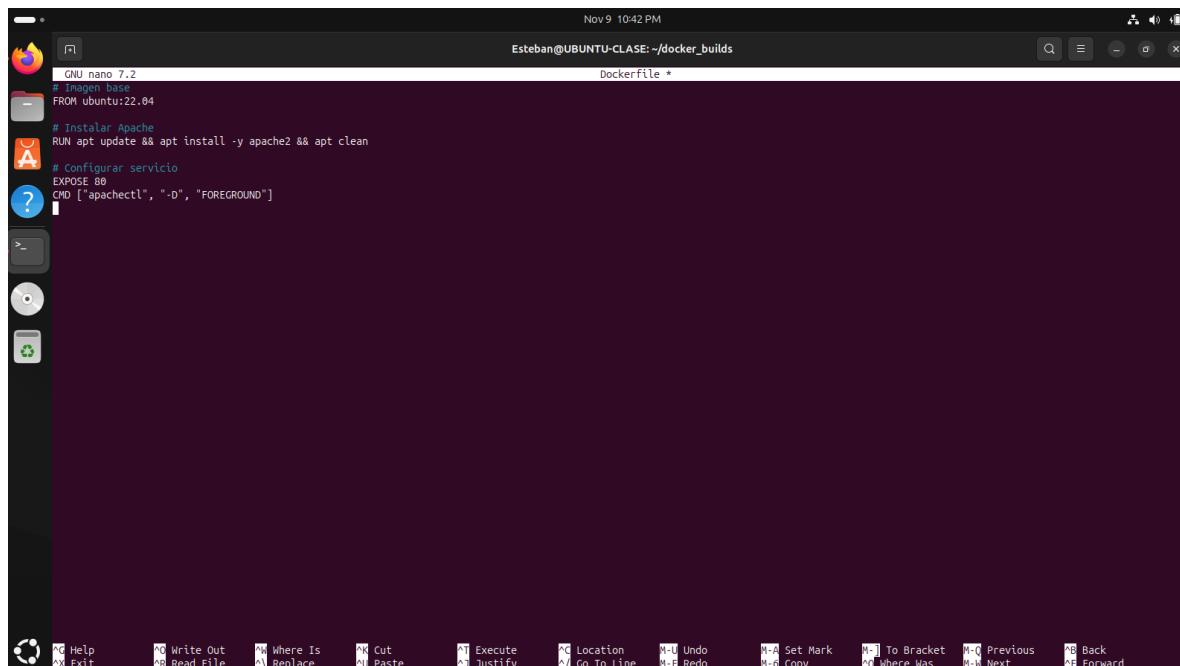
3. Crear el Dockerfile para Apache

Preparar el espacio de trabajo para construir la imagen del servicio Apache. Se crea una carpeta dedicada y se accede a ella para organizar los archivos de construcción:

- mkdir -p ~/docker_builds/apache
- cd ~/docker_builds/apache

```
Esteban@UBUNTU-CLASE:~/docker_builds$ sudo nano Dockerfile
Esteban@UBUNTU-CLASE:~/docker_builds$ ls
apache Dockerfile mysql nginx
Esteban@UBUNTU-CLASE:~/docker_builds$
```

Con esto se garantiza una estructura ordenada por servicio, facilitando la gestión de Dockerfiles, recursos y versiones.



The screenshot shows a terminal window titled "Dockerfile *". The command "ls" has been run, showing files named "apache", "Dockerfile", "mysql", and "nginx". The "Dockerfile" file is open in a nano editor. The content of the Dockerfile is as follows:

```
GNU nano 7.2
# Instalar base
FROM ubuntu:22.04

# Instalar Apache
RUN apt update && apt install -y apache2 && apt clean

# Configurar servicio
EXPOSE 80
CMD ["apachectl", "-D", "foreground"]
```

- Imagen base: se utiliza ubuntu:22.04 como punto de partida porque proporciona un sistema limpio, estable y ampliamente soportado, lo que facilita reproducibilidad y mantenimiento.

- Instalación del servicio: se instalan los paquetes de Apache necesarios dentro de la imagen para disponer del servidor web y sus utilidades básicas, garantizando que la aplicación pueda atender solicitudes HTTP.
- Exposición del puerto: se expone el puerto 80, que es el estándar para tráfico web HTTP, permitiendo que el contenedor reciba conexiones desde el host o desde otros servicios de la red.
- Proceso en primer plano: se configura el arranque de Apache en primer plano para que el proceso principal del contenedor permanezca activo; en Docker, si el proceso PID 1 termina, el contenedor se detiene, por lo que ejecutar Apache en foreground asegura su disponibilidad continua.

```
Esteban@UBUNTU-CLASE:~/docker_builds/apache$ ls -l ~/docker_builds/apache
total 4
-rw-r--r-- 1 root root 176 Nov 9 22:48 Dockerfile
Esteban@UBUNTU-CLASE:~/docker_builds/apache$ cd ~/docker_builds/apache
sudo docker build -t apache_custom .
[+] Building 73.5s (6/6) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 215B
=> [internal] load metadata for docker.io/library/ubuntu:22.04
=> => transferring context: 2B
=> [1/2] FROM docker.io/library/ubuntu:22.04@sha256:09506232a8004baa32c47d68f1e5c307d648fdd59f5e7eaa42aaef87914100db3
=> => resolve docker.io/library/ubuntu:22.04@sha256:09506232a8004baa32c47d68f1e5c307d648fdd59f5e7eaa42aaef87914100db3
=> => sha256:09506232a8004baa32c47d68f1e5c307d648fdd59f5e7eaa42aaef87914100db3 6.69kB / 6.69kB
=> => sha256:4cb708d50443fc4463ff1f9360c03ca46512e4fd8fd97c5ce7e69c8758924575 424B / 424B
=> => sha256:392fa14ddd09da9a5c3d26948ff81c494424035b755d01b84ab12d92127433 2.30kB / 2.30kB
=> => sha256:a6ecca94c8104c8e90d3f9ebe59c2b3a02b20aad3d985e31c7cd09ea104c447 29.54MB / 29.54MB
=> => extracting sha256:a6ecca94c8104c8e90d3f9ebe59c2b3a02b20aad3d985e31c7cd09ea104c447
[2/2] RUN apt update && apt install -y apache2 && apt clean
=> exporting to image
=> => exporting layers
=> => writing image sha256:c81c03d2d4698a18140ba4fcf022632ec8c9e069514a6f764f792a365f98fc76
=> => naming to docker.io/library/apache_custom
Esteban@UBUNTU-CLASE:~/docker_builds/apache$
```

Construcción de la imagen del servicio Apache. Se posiciona el contexto de compilación en la carpeta del proyecto y se ejecuta la construcción etiquetando la imagen para su fácil identificación local.

- Ubicar el contexto:
 - cd ~/docker_builds/apache
- Construir y etiquetar:
 - sudo docker build -t apache_custom.

Con esta acción se genera una imagen local llamada apache_custom, a partir del Dockerfile en el directorio actual, lista para ser usada en contenedores.



```
Esteban@UBUNTU-CLASE:~/docker_builds/apache$ cat Dockerfile
# Imagen base
FROM ubuntu:22.04

# Instalar Apache
RUN apt update && apt install -y apache2 && apt clean

# Configurar servicio
EXPOSE 80
CMD ["apachectl", "-D", "FOREGROUND"]

Esteban@UBUNTU-CLASE:~/docker_builds/apache$ sudo docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
apache_custom   latest   c81c03d2d469  4 minutes ago  256MB
<none>          <none>   fbfcdad71cbde  2 days ago   64.6MB
nginx_custom    latest   50f7e8619bd0  2 days ago   52.8MB
phpmyadmin      latest   5d0eabeabcf5  5 days ago   575MB
nginx           latest   d261fd19cb63  6 days ago   152MB
mariadb          latest   dfbea441e6fc  3 months ago  330MB
busybox          latest   08ef35a1c3f0  13 months ago  4.43MB

Esteban@UBUNTU-CLASE:~/docker_builds/apache$
```

Verifica que el archivo quedó bien y la imagen con el comando **cat Dockerfile** y **sudo docker images**

4. Crear el Dockerfile para MySQL

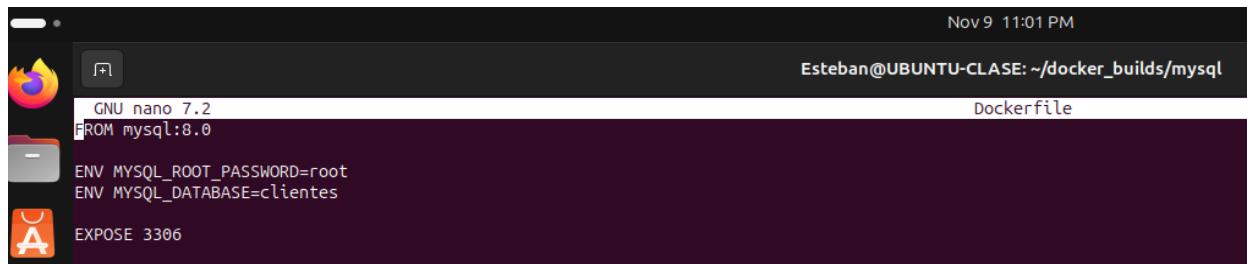
```
Esteban@UBUNTU-CLASE:~$ mkdir -p ~/docker_builds/mysql
cd ~/docker_builds/mysql
```

5. Crear el Dockerfile para MySQL

Preparar el espacio de trabajo para construir la imagen del servicio MySQL. Se crea una carpeta dedicada y se accede a ella para organizar los archivos de construcción:

- mkdir -p ~/docker_builds/mysql
- cd ~/docker_builds/mysql

```
Esteban@UBUNTU-CLASE:~/docker_builds/mysql$ sudo nano Dockerfile
Esteban@UBUNTU-CLASE:~/docker_builds/mysql$ ls
Dockerfile
```



```
Nov 9 11:01 PM
Esteban@UBUNTU-CLASE: ~/docker_builds/mysql Dockerfile
GNU nano 7.2
FROM mysql:8.0
ENV MYSQL_ROOT_PASSWORD=root
ENV MYSQL_DATABASE=clientes
EXPOSE 3306
```

Crear el archivo Dockerfile para el servicio de base de datos MySQL. Se genera el archivo y se define una imagen basada en MySQL 8 con variables de entorno mínimas y el puerto estándar expuesto.

- **Abrir/crear el archivo:**

`sudo nano Dockerfile`

- **Contenido del Dockerfile:**

```
FROM mysql:8.0
ENV MYSQL_ROOT_PASSWORD=root
ENV MYSQL_DATABASE=clientes
EXPOSE 3306
```

Este Dockerfile usa la imagen oficial mysql:8.0, establece la contraseña del usuario root y crea la base de datos inicial indicada mediante variables de entorno, además de exponer el puerto 3306 para conexiones al servidor MySQL.

```
Esteban@UBUNTU-CLASE:~/docker_builds/mysql$ cat Dockerfile
FROM mysql:8.0

ENV MYSQL_ROOT_PASSWORD=root
ENV MYSQL_DATABASE=clientes

EXPOSE 3306
Esteban@UBUNTU-CLASE:~/docker_builds/mysql$ █
```

Verifica que el archivo quedó bien y la imagen con el comando **cat Dockerfile** y **sudo docker images**



```

Estephan@UBUNTU-CLASE:~/docker/builds/mysql$ sudo docker build -t mysql_custom .
[+] Building 28.3s (5/5) FINISHED
--> [internal] load build definition from Dockerfile
--> transferring dockerfile: 123B
--> [internal] load metadata for docker.io/library/mysql:8.0
--> [internal] load .dockerignore
--> transferring context: 2B
--> resolve docker.io/library/mysql:8.0@sha256:f37951fc3753a6a22d0c7bf6978c5e5fefcf631814d98c502524f98ae52b21
--> sha256:53cc03981ba050aa4422b192a01dd031465330157@abb07954849ca5_0.04kB
--> sha256:1178dbae06755997113cb0d31f104ab99680947fb0b65725fbfa322a_0.74kB / 0.74kB
--> sha256:1f7951fc3753a6a22d0c7bf6978c5e5fefcf631814d98c502524f98ae52b21_2.06kB / 2.06kB
--> sha256:103102c60a00422d0c7bf6978c5e5fefcf631814d98c502524f98ae52b21_49.50MB / 49.50MB
--> sha256:a2ed1082d9e2d4196202fd3fc3dc314199a67410fad3208e19605898657ce50_783.56kB
--> sha256:47fe344dfad6f0620570919875ce764acd335e2f744c249447608c940b24a6a_885 kB / 885 kB
--> sha256:47fe344dfad6f0620570919875ce764acd335e2f744c249447608c940b24a6a_885 kB / 885 kB
--> sha256:c1eef70ed8e709da9e0f6918406f73830f37353c5fd1bf17efcf8e0ca307_0.02kB / 0.02kB
--> sha256:47fe344dfad6f0620570919875ce764acd335e2f744c249447608c940b24a6a_2.06kB / 2.06kB
--> sha256:2a2d524403ab5a16280c1bce5ee560d00a01deaf39a1e396050b0d01c2b5591f7_334 kB / 334 kB
--> sha256:47fe344dfad6f0620570919875ce764acd335e2f744c249447608c940b24a6a_49.90MB / 49.90MB
--> sha256:fdcf97503d442a26171ecf71854acc903bce74937321b557c5c97c95a190c0_3178 kB / 3178 kB
--> sha256:1f302d442a26171ecf71854acc903bce74937321b557c5c97c95a190c0_127.87MB / 127.87MB
--> extracttiny sha256:0231a12c62a6ce5af724930e7e994cea33b2c2db5f1a0035501015efdf3ed
--> sha256:1370d510f7651d7233211508a304b0195680370b42c3f7913a13126933c1e4ed_5.33kB / 5.33kB
--> sha256:d087104a4d9516175a50b5bd8d0a060a209404ab0b620d1c95e07b033317_1218 kB / 1218 kB
--> extracting sha256:47fe344dfad6f0620570919875ce764acd335e2f744c249447608c940b24a6a_0.05 kB
--> extracting sha256:a2ed1082d9e2d4196202fd3dc314199a67410fad3208e19605898657ce50
--> extracting sha256:c1eef70ed8e709da9e0f6918406f73830f37353c5fd1bf17efcf8e0ca307
--> extracting sha256:47fe344dfad6f0620570919875ce764acd335e2f744c249447608c940b24a6a_0.05 kB
--> exporting to image
--> exporting layers
--> writing Image sha256:008db2ff4f9e0042fadd47e76672b44c6a94102c71e47c47df95062fe0cff5001
--> naming to docker.io/library/mysql_custom

1 warning found (use docker --debug to expand).
- Secrets used in argon2env do not use ENV instructions for sensitive data (ENV "MYSQL_ROOT_PASSWORD") (line 3)
Estephan@UBUNTU-CLASE:~/docker/builds/mysql$ 

```

Verificación del Dockerfile y construcción de la imagen personalizada. Se valida que el archivo Dockerfile contenga exactamente las instrucciones previstas y, posteriormente, se construye una imagen personalizada basada en MySQL 8.0. La advertencia de seguridad observada se debe al uso de variables de entorno con datos sensibles dentro de la imagen, por lo que se documenta su causa y la forma recomendada de mitigación.

- **Verificar contenido del Dockerfile:** cat Dockerfile
 - **Debe coincidir con:**

FROM mysql:8.0

ENV MYSQL_ROOT_PASSWORD=root

ENV MYSQL DATABASE=clientes

EXPOSE 3306

- **Construir la imagen:** sudo docker build -t mysql_custom
 - **Qué significa la advertencia:** Docker desaconseja definir secretos de una imagen con ENV, porque quedan almacenados en las capas y pueden si alguien accede a la imagen o a su historial. El mensaje “Do not use ARG or sensitive data” indica ese riesgo y sugiere usar mecanismos alternativos de credenciales en tiempo de ejecución.

- **Opción A (recomendada en escenarios con orquestación):** usar Docker Secrets para inyectar la contraseña en tiempo de ejecución, evitando guardarla en la imagen. Esto permite referenciar un archivo secreto dentro del contenedor en lugar de un valor en texto plano.

- **Opción B (escenario simple sin Swarm):** No poner la contraseña en el Dockerfile. Definirla al ejecutar el contenedor mediante una variable de entorno externa o un archivo montado, manteniendo el secreto fuera del control de versiones.

- **Opción C (endurecimiento adicional de MySQL):** Se utiliza el MYSQL_RANDOM_ROOT_PASSWORD para que la imagen genere una contraseña aleatoria en la inicialización y luego rotarla de forma segura.

Con estas modificaciones, el Dockerfile resulta adecuado para propósitos académicos y, al mismo tiempo, se deja constancia de una práctica segura para el manejo de credenciales en entornos de producción, evitando su inclusión directa en la imagen.

```
Esteban@UBUNTU-CLASE:~/docker_builds/mysql$ cat Dockerfile
FROM mysql:8.0

ENV MYSQL_ROOT_PASSWORD=root
ENV MYSQL_DATABASE=clientes

EXPOSE 3306

Esteban@UBUNTU-CLASE:~/docker_builds/mysql$ sudo docker images
REPOSITORY          TAG        IMAGE ID      CREATED       SIZE
apache_custom       latest     c81c03d2d469   29 minutes ago  256MB
<none>              <none>    fbfcd71cbde    2 days ago    64.6MB
nginx_custom        latest     50f7e8619bd0    2 days ago    52.8MB
phpmyadmin          latest     5d0eabeabcf5    5 days ago    575MB
nginx               latest     d261fd19cb63    6 days ago    152MB
mysql_custom        latest     00db2ff4f9e0    2 weeks ago   783MB
mariadb             latest     dfbea441e6fc    3 months ago   330MB
busybox             latest     08ef35a1c3f0    13 months ago  4.43MB

Esteban@UBUNTU-CLASE:~/docker_builds/mysql$
```

Verifica que el archivo quedó bien y la imagen con el comando **cat Dockerfile** y **sudo docker images**

6. Crear el Dockerfile para Nginx

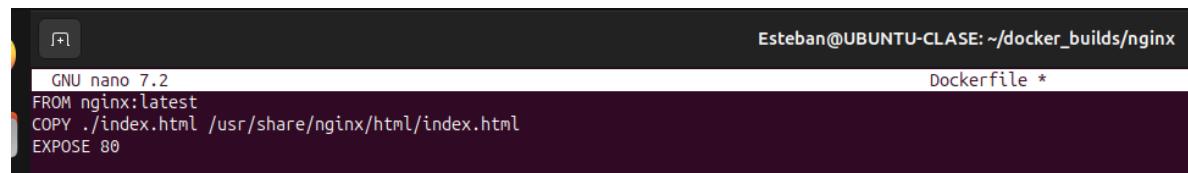
```
Esteban@UBUNTU-CLASE:~$ mkdir -p ~/docker_builds/nginx
cd ~/docker_builds/nginx
Esteban@UBUNTU-CLASE:~/docker_builds/nginx$
```

7. Crear el archivo index.html dentro de la carpeta:

Preparar el espacio de trabajo para construir la imagen del servicio Nginx. Se crea una carpeta dedicada y se accede a ella para organizar los archivos de construcción:

- mkdir -p ~/docker_builds/Nginx
- cd ~/docker_builds/Nginx

```
Esteban@UBUNTU-CLASE:~/docker_builds/nginx$ sudo nano Dockerfile
Esteban@UBUNTU-CLASE:~/docker_builds/nginx$ ls
Dockerfile
Esteban@UBUNTU-CLASE:~/docker_builds/nginx$
```



```
GNU nano 7.2
FROM nginx:latest
COPY ./index.html /usr/share/nginx/html/index.html
EXPOSE 80
```

- **Creación del Dockerfile para Nginx y página de prueba:** Se define una imagen basada en Nginx que publica el puerto 80 y despliega una página HTML sencilla para verificar el correcto funcionamiento del servicio.

- **Contenido del Dockerfile:**

```
FROM nginx:latest
COPY ./index.html /usr/share/nginx/html/index.html
EXPOSE 80
```

- **Crear el archivo index.html en la carpeta del proyecto:** echo "<h1>Servidor Nginx funcionando correctamente</h1>" > ~/docker_builds/nginx/index.html

Con esta configuración, al construir y ejecutar la imagen, Nginx servirá la página index.html en el puerto 80, permitiendo comprobar visualmente que el contenedor responde como se espera.

```
Esteban@UBUNTU-CLASE:~/docker_builds/nginx$ echo "<h1>Servidor Nginx funcionando correctamente</h1>" > ~/docker_builds/nginx/index.html
Esteban@UBUNTU-CLASE:~/docker_builds/nginx$ cd ~/docker_builds/nginx
sudo docker build -t nginx_custom .
[+] Building 0.6s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 116B
=> [internal] load metadata for docker.io/library/nginx:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 87B
=> [1/2] FROM docker.io/library/nginx:latest
=> [2/2] COPY ./index.html /usr/share/nginx/html/index.html
=> exporting to image
=> => exporting layers
=> => writing image sha256:8b0d443e3963780bc490b059bd087a295d10d4dc6c82473cda0e81d433b0b496
=> => naming to docker.io/library/nginx_custom
Esteban@UBUNTU-CLASE:~/docker_builds/nginx$
```

9. Ejecutar los contenedores con volúmenes RAID/LVM

```
Esteban@UBUNTU-CLASE:~/docker_builds/nginx$ sudo docker run -d --name cont_apache -p 8080:80 -v /mnt/apache_vol:/var/www/html apache_custom
b31ee0dca18817b66f77fc88161d8fffc8668915de86fe5bd2a64310770b86377
Esteban@UBUNTU-CLASE:~/docker_builds/nginx$ sudo docker run -d --name cont_mysql -e MYSQL_ROOT_PASSWORD=root -v /mnt/mysql_vol:/var/lib/mysql mysql_custom
814866e9958811251f800ff9ae78dbe6aa8146e1c911abc2aa24c9f489470f
Esteban@UBUNTU-CLASE:~/docker_builds/nginx$ sudo docker run -d --name cont_nginx -p 8081:80 -v /mnt/nginx_vol:/usr/share/nginx/html nginx_custom
d3a8ddfe9b2ad424a4c3ba4710d972a545b8af144704ef98f88be5461f7fd071
Esteban@UBUNTU-CLASE:~/docker_builds/nginx$
```

Ejecución de los contenedores con volúmenes y puertos asignados: Se inician los servicios Apache, MySQL y Nginx en modo desatendido, mapeando puertos y montando los volúmenes persistentes para sus datos y contenidos.

- ❖ **Apache (puerto 8080 → 80, contenido en /mnt/apache_vol):** sudo docker run -d --name cont_apache -p 8080:80 -v /mnt/apache_vol:/var/www/html apache_custom
- ❖ **MySQL (datos en /mnt/mysql_vol; nota: la contraseña se define aquí solo con fines académicos):** sudo docker run -d --name cont_mysql -e MYSQL_ROOT_PASSWORD=root -v /mnt/mysql_vol:/var/lib/mysql mysql_custom
- ❖ **Nginx (puerto 8081 → 80, contenido en /mnt/nginx_vol):** sudo docker run -d --name cont_nginx -p 8081:80 -v /mnt/nginx_vol:/usr/share/nginx/html nginx_custom

Con estos comandos, cada servicio queda aislado en su contenedor, con datos persistentes en los volúmenes LVM montados y accesibles desde los puertos publicados del host.

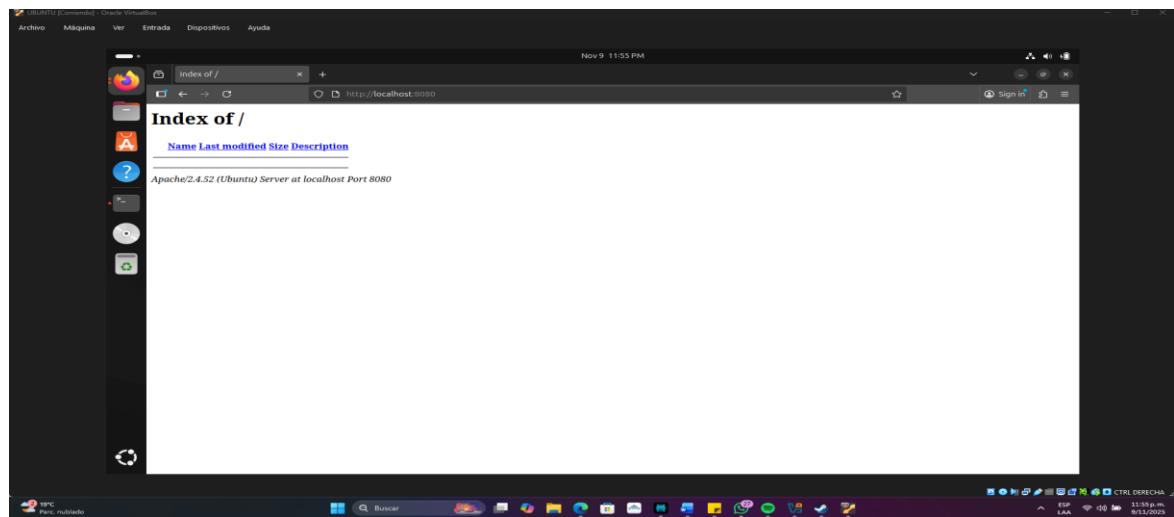
10. Pruebas de funcionamiento

- **Verificación de contenedores en ejecución:** Comando utilizado

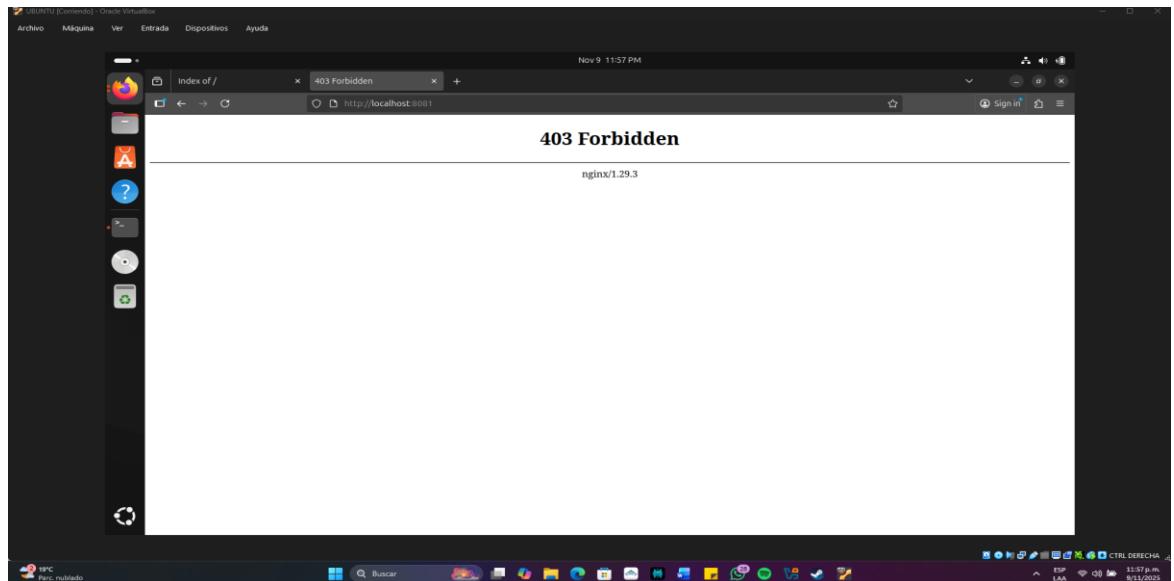
`sudo Docker ps`

```
Esteban@UBUNTU-CLASE:~/docker_builds/nginx$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
d3a0ddfe9bb2a nginx_custom "/docker-entrypoint..." 7 minutes ago Up 7 minutes 0.0.0.0:8081->80/tcp, [::]:8081->80/tcp cont_nginx
814066e99588 mysql_custom "docker-entrypoint.s..." 7 minutes ago Up 7 minutes 3306/tcp, 33060/tcp cont_mysql
b31ee0dca188 apache_custom "apachectl -D FOREGR..." 7 minutes ago Up 7 minutes 0.0.0.0:8080->80/tcp, [::]:8080->80/tcp cont_apache
Esteban@UBUNTU-CLASE:~/docker_builds/nginx$
```

- **Servidor utilizando Apache:** Dirección que vamos a emplear para realizar la prueba del funcionamiento localhost:8080



- **Servidor utilizando Nginx:** Dirección que vamos a emplear para realizar la prueba del funcionamiento localhost:8081



- **Servidor de MySQL:**
 1. **Accede a la base de datos:** Empleando el comando sudo docker exec -it cont_mysql mysql -u root -p# contraseña: root



```
Esteban@UBUNTU-CLASE:~$ sudo docker exec -it cont_mysql mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.44 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| clientes |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.00 sec)

mysql> USE clientes;
Database changed
mysql> CREATE TABLE prueba (id INT, nombre VARCHAR(50));
Query OK, 0 rows affected (0.08 sec)

mysql> INSERT INTO prueba VALUES (1, 'persistencia');
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM prueba;
+---+---+
| id | nombre |
+---+---+
| 1 | persistencia |
+---+---+
1 row in set (0.00 sec)

mysql> ■
```

- **Base de datos de MySQL utilizando phpMyAdmin:**
- 2. **Crea y ejecuta el contenedor phpMyAdmin:** Comandos utilizados para la creación del contenedor

sudo docker run -d \ --name phpmyadmin \



```
-e PMA_HOST=cont_mysql \
-e PMA_USER=root \
-e PMA_PASSWORD=root \
-p 8082:80 \
--link cont_mysql:db \
phpmyadmin/phpmyadmin
```

```
Esteban@UBUNTU-CLASE:~$ sudo docker run -d \
--name phpmyadmin \
-e PMA_HOST=cont_mysql \
-e PMA_USER=root \
-e PMA_PASSWORD=root \
-p 8082:80 \
--link cont_mysql:db \
phpmyadmin/phpmyadmin
Unable to find image 'phpmyadmin/phpmyadmin:latest' locally
latest: Pulling from phpmyadmin/phpmyadmin
8c7716127147: Pull complete
24403a1f6855: Pull complete
e1cf44d6017a: Pull complete
2489d5e860a7: Pull complete
0248257cbd51: Pull complete
dd53cf9bf4cf: Pull complete
a139c2f3234a: Pull complete
6571cfdbe5b2: Pull complete
8d83c968ca9a: Pull complete
fddb92e888a7: Pull complete
749b92ea0995: Pull complete
4eed3454c20c: Pull complete
00ef78e422f0: Pull complete
004f06ab2f6c: Pull complete
4f4fb700ef54: Pull complete
273a09302c4a: Pull complete
a8c81cca4b8b: Pull complete
271012172ea0: Pull complete
0befb457381d: Pull complete
30b083700c69: Pull complete
b5a8243c1a5e: Pull complete
Digest: sha256:42a200db07b4e70fbf32c594ad4521cf16399b8e54bbb5adceae98e7566dfbeb
Status: Downloaded newer image for phpmyadmin/phpmyadmin:latest
b2af5198361075b4c1aa68afa61e3833b4ac15f2b64e8cb91bcd57c6af53aace
Esteban@UBUNTU-CLASE:~$ █
```

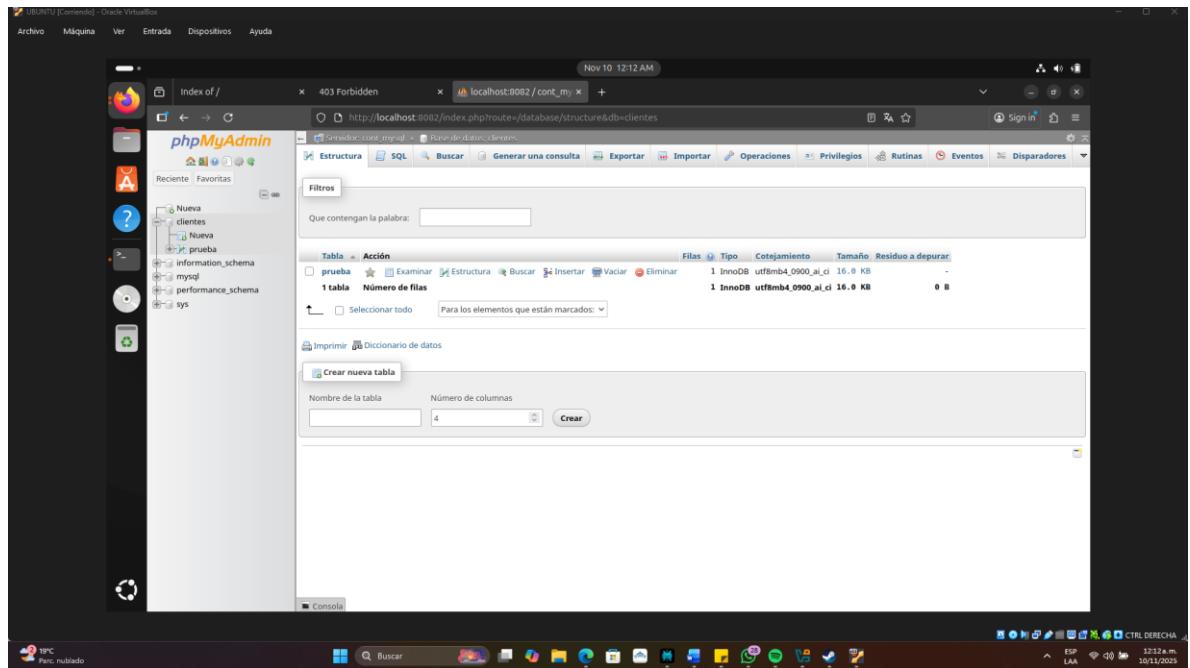
- Verificar que estén funcionando los contenedores:



```
Esteban@UBUNTU-CLASE:~$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
b2af51983610 phpmyadmin/phpmyadmin "/docker-entrypoint..." 34 seconds ago Up 33 seconds 0.0.0.0:8082->80/tcp, [::]:8082->80/tcp phpmyadmin
d3a8ddfe9b2a nginx_custom "/docker-entrypoint..." 25 minutes ago Up 25 minutes 0.0.0.0:8081->80/tcp, [::]:8081->80/tcp cont_nginx
814066e99588 mysql_custom "docker-entrypoint.s..." 25 minutes ago Up 25 minutes 3306/tcp, 33060/tcp cont_mysql
b31ee0dca188 apache_custom "apachectl -D FOREGR..." 25 minutes ago Up 25 minutes 0.0.0.0:8080->80/tcp, [::]:8080->80/tcp cont_apache
Esteban@UBUNTU-CLASE:~$
```

- **Interfaz de phpMyAdmin:** Se accede desde el navegador utilizando la url <http://localhost:8082> y validamos la base de datos de cliente que creamos anteriormente utilizando nuestra terminal

The screenshot shows a Linux desktop environment with a dark theme. A Firefox browser window is open to the phpMyAdmin interface at <http://localhost:8082>. The browser title bar says 'localhost:8082 / cont_my'. The phpMyAdmin dashboard is visible, showing configuration sections like 'Configuraciones generales' and 'Configuraciones de apariencia', and detailed sections for the database server ('Servidor de base de datos') and web server ('Servidor web'). The desktop background is a dark green color, and the taskbar at the bottom has icons for various applications like file manager, terminal, and system monitoring.



Para visualizar la base de datos MySQL en entorno web utilicé phpMyAdmin, ejecutado como contenedor Docker vinculado al contenedor MySQL, lo que demuestra la integración entre servicios y la virtualización de aplicaciones con volúmenes persistentes.

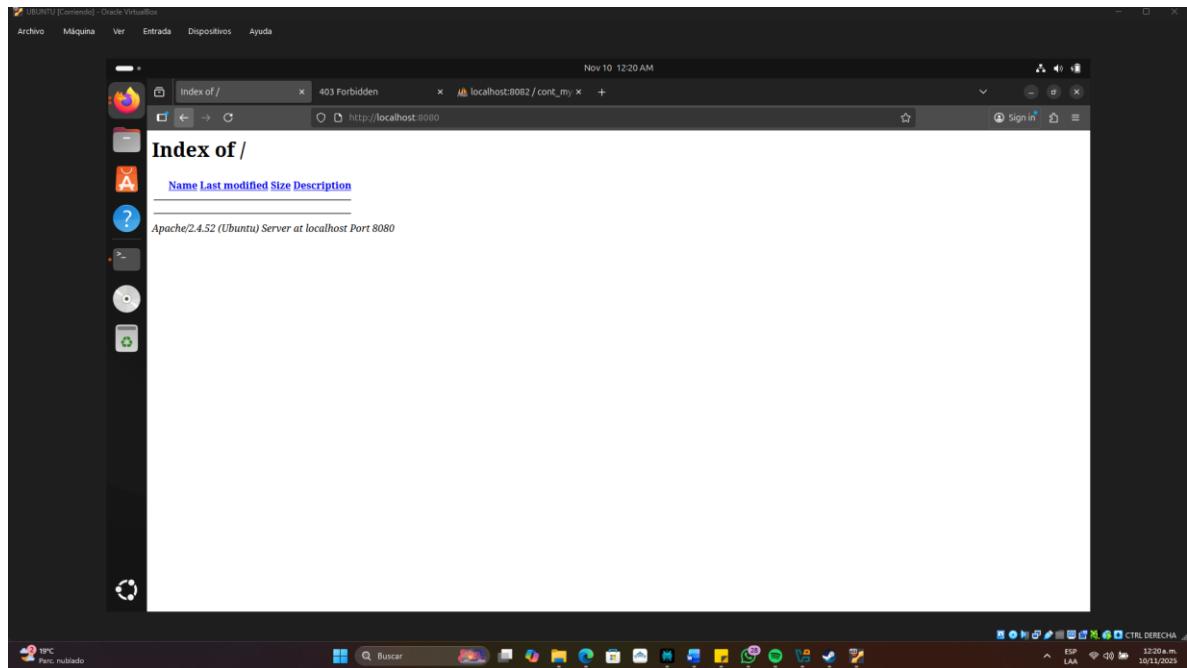
3. Prueba de persistencia Apache:

- Crea o modifica un archivo dentro del volumen, por ejemplo:

```
Esteban@UBUNTU-CLASE:~$ echo "<h1>Prueba de persistencia Apache</h1>" | sudo tee /mnt/apache_vol/index.html
<h1>Prueba de persistencia Apache</h1>
Esteban@UBUNTU-CLASE:~$
```

- **Reinicia el contenedor:** sudo docker restart cont_apache

- ❖ Primera captura antes de reiniciar el contenedor:



❖ Segunda captura después de reiniciarlo

```
Esteban@UBUNTU-CLASE:~$ sudo docker restart cont_apache
Error response from daemon: Cannot restart container cont_apache: permission denied
Esteban@UBUNTU-CLASE:~$ sudo docker logs cont_apache
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
Esteban@UBUNTU-CLASE:~$
```

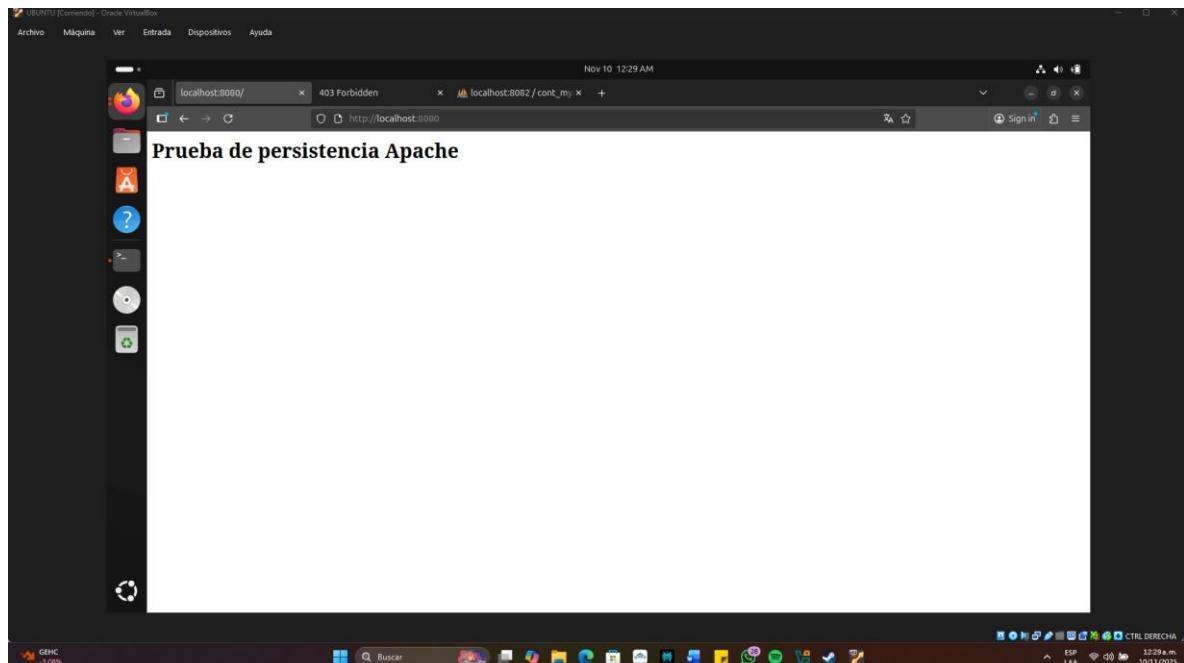
- **Observación técnica – Mensaje AH00558 en Apache durante la ejecución del contenedor Apache, el registro de Docker mostró el siguiente mensaje:**

AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message.

- **Explicación:** Este mensaje no representa un error. Indica que Apache no tiene configurado un nombre de dominio (ServerName), por lo cual usa la dirección IP interna del contenedor (172.17.0.2) como nombre por defecto.

- **Solución aplicada (opcional):** Se puede eliminar agregando la línea:

ServerName localhost en el archivo /etc/apache2/apache2.conf. Sin embargo, no afecta el funcionamiento del servicio, por lo que se mantuvo como observación en la bitácora.



- **Prueba de persistencia utilizando MySQL**

1. Ingresamos al contenedor: **sudo docker exec -it cont_mysql mysql -**

u root -p

2. **Nos conectamos a la base de datos:** Utilizando la contraseña de root

y el comando use clientes;

3. **Crea una nueva tabla:**

```
CREATE TABLE persistencia2 (
    id INT PRIMARY KEY,
    descripcion VARCHAR(100)
);
```

```
Esteban@UBUNTU-CLASE:~$ sudo docker exec -it cont_mysql mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 30
Server version: 8.0.44 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE clientes;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> CREATE TABLE persistencia2 (
    ->     id INT PRIMARY KEY,
    ->     descripcion VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.08 sec)

mysql>
```

4. Creamos un dato de prueba: INSERT INTO persistencia2

VALUES (1, 'Segunda prueba de persistencia con RAID y LVM');

```
mysql> INSERT INTO persistencia2 VALUES (1, 'Segunda prueba de persistencia con RAID y LVM');
Query OK, 1 row affected (0.03 sec)

mysql> ■
```

5. Se verifica que se insertó correctamente: SELECT * FROM persistencia2;

```
mysql> SELECT * FROM persistencia2;
+----+
| id | descripcion |
+----+
| 1  | Segunda prueba de persistencia con RAID y LVM |
+----+
1 row in set (0.00 sec)

mysql>
```

6. Prueba de persistencia: Utilizamos los siguientes comandos después de salirnos de la base de datos utilizando exit

```
sudo docker restart cont_mysql
sudo docker exec -it cont_mysql mysql -u root -p
```

USE clientes;

SELECT * FROM persistencia2;

```
Esteban@UBUNTU-CLASE:~$ sudo docker restart cont_mysql
sudo docker exec -it cont_mysql mysql -u root -p
USE empresa;
SELECT * FROM persistencia2;
Error response from daemon: Cannot restart container cont_mysql: permission denied
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 33
Server version: 8.0.44 MySQL Community Server - GPL

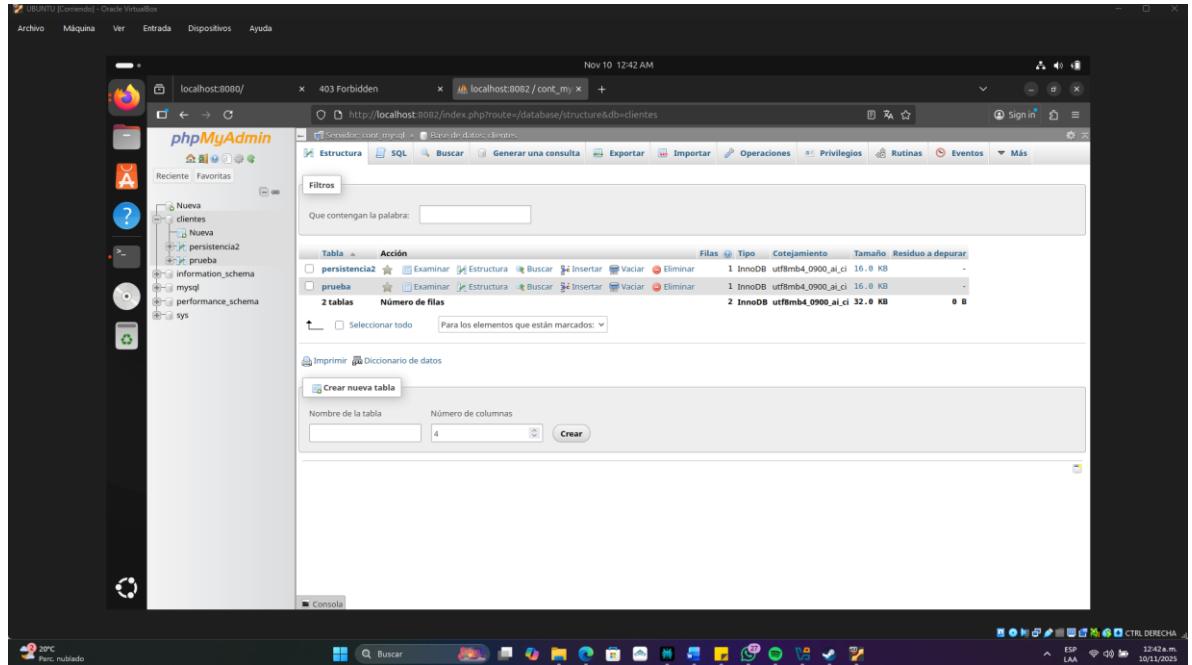
Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> |
```

- Se verifica en phpMyAdmin para ver si esta la persistencia al momento de hacer los pasos anteriores



The screenshot shows the phpMyAdmin interface running on a Windows desktop. The browser window title is "localhost:8082 / cont_my". The left sidebar shows databases: Nueva, clientes, persistencia2, prueba, mysql, information_schema, and sys. The 'persistencia2' database is selected. The main panel displays the structure of the 'clientes' table. The table has two rows:

Tabla	Acción	Filas	Tipo	Colejamiento	Tamaño	Residuo a depurar
persistencia2	Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	utf8mb4_0900_ai_ci	16.0 KB	
prueba	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_0900_ai_ci	16.0 KB	
	Número de filas	2 tablas			32.0 KB	0 B

Below the table, there is a 'Crear nueva tabla' (Create new table) form with 'Nombre de la tabla' (Table name) set to '4'.

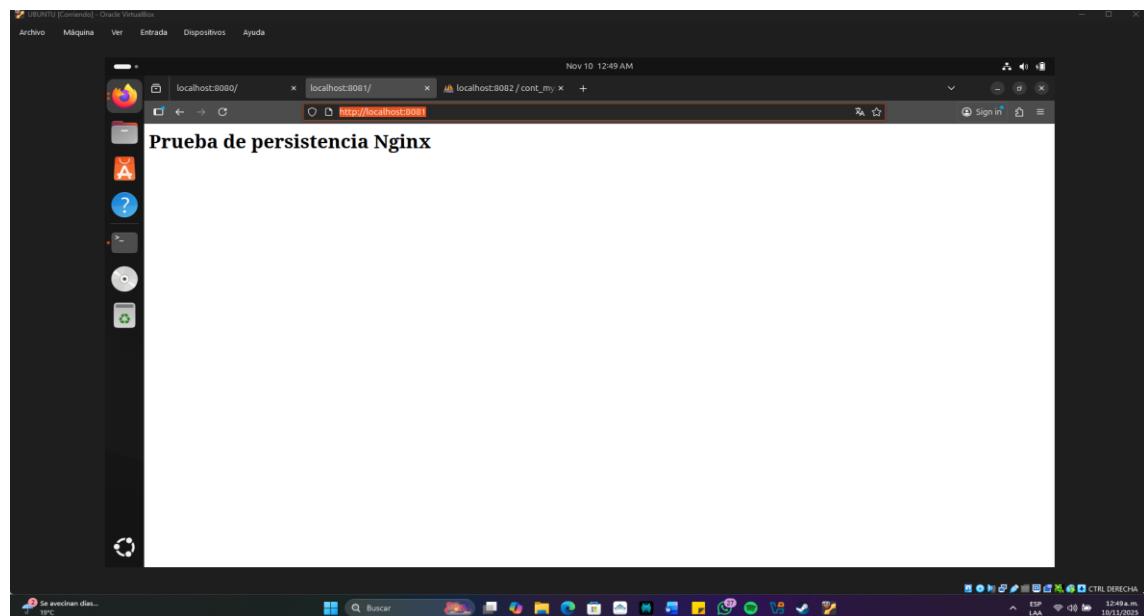
4. **Prueba de persistencia en Nginx: Objetivo:** Comprobar que los archivos del servidor web Nginx se mantienen tras reiniciar el contenedor.

1. **Entrar al volumen: Vamos a modificar el archivo directamente en el volumen:** echo "<h1>Prueba de persistencia Nginx</h1>" | sudo tee /mnt/nginx_vol/index.html

```
Esteban@UBUNTU-CLASE:~$ echo "<h1>Prueba de persistencia Nginx</h1>" | sudo tee /mnt/nginx_vol/index.html
<h1>Prueba de persistencia Nginx</h1>
Esteban@UBUNTU-CLASE:~$
```

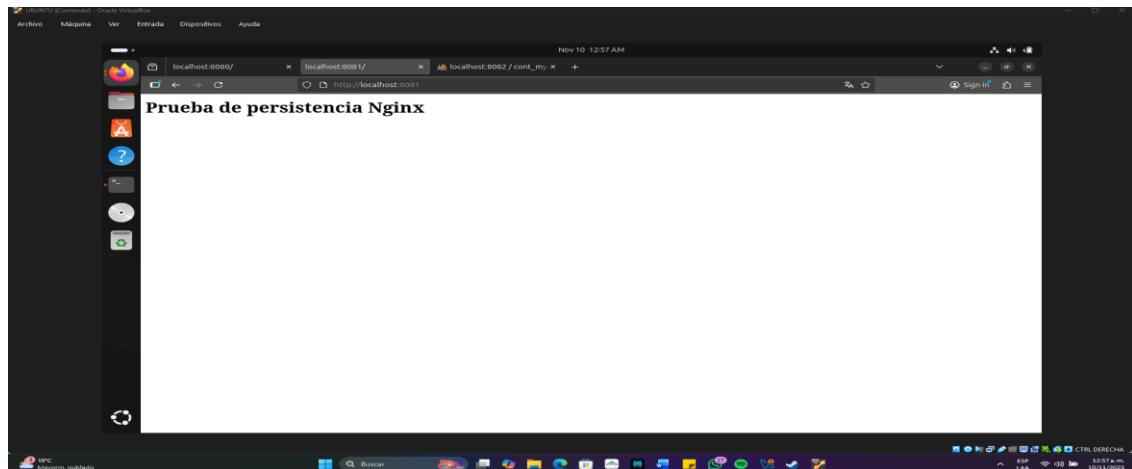
2. **Verifica desde el navegador o terminal:**

Captura con persistencia



3. **Reinicia el contenedor:** Comando que utilizaremos sudo docker restart cont_nginx

```
Esteban@UBUNTU-CLASE:~$ sudo docker restart cont_nginx
Error response from daemon: Cannot restart container cont_nginx: permission denied
```



1. Resultado Final Fase 3

Servicio	Contenedo	Puerto	Volumen (LVM)	Estado
Apache	cont_apach	8080	/mnt/apache_vol	<input checked="" type="checkbox"/>
MySQL	cont_mysql	intern o 3306	/mnt/mysql_vol	<input checked="" type="checkbox"/>
Nginx	cont_nginx	8081	/mnt/nginx_vol	<input checked="" type="checkbox"/>

2. Fase 5 — Implementación con Podman

Objetivo: Ejecutar los mismos contenedores (Apache, MySQL y Nginx) usando Podman, demostrando compatibilidad con Docker y persistencia en los volúmenes RAID/LVM.

1. Instalación de Podman



```
Esteban@UBUNTU-CLASE:~$ sudo apt update
[sudo] password for Esteban:
Hit:1 http://co.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://co.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://co.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
5 packages can be upgraded. Run 'apt list --upgradable' to see them.
Esteban@UBUNTU-CLASE:~$ sudo apt install -y podman
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  liblvm19
Use 'sudo apt autoremove' to remove it.
Suggested packages:
  containers-storage
The following NEW packages will be installed:
  podman
0 upgraded, 1 newly installed, 0 to remove and 5 not upgraded.
Need to get 13.4 MB of archives.
After this operation, 43.6 MB of additional disk space will be used.
Get:1 http://co.archive.ubuntu.com/ubuntu noble-updates/universe amd64 podman amd64 4.9.3+ds1-1ubuntu0.2 [13.4 MB]
Fetched 13.4 MB in 2s (7,066 kB/s)
Selecting previously unselected package podman.
(Reading database ... 194009 files and directories currently installed.)
Preparing to unpack .../podman_4.9.3+ds1-1ubuntu0.2_amd64.deb ...
Unpacking podman (4.9.3+ds1-1ubuntu0.2) ...
Setting up podman (4.9.3+ds1-1ubuntu0.2) ...
Processing triggers for man-db (2.12.0-4build2) ...
Esteban@UBUNTU-CLASE:~$
```

Comandos utilizados para realizar la instalación de podman en nuestra máquina virtual:

```
sudo apt update
sudo apt install -y podman
```

2. Comprobación de compatibilidad con Docker

```
Esteban@UBUNTU-CLASE:~$ podman --version
podman version 4.9.3
Esteban@UBUNTU-CLASE:~$ alias docker=podman
Esteban@UBUNTU-CLASE:~$
```

Podman puede ejecutar los mismos comandos que Docker: alias docker=podman, además, esto permite usar exactamente los mismos docker run, docker ps, docker build, etc., pero con Podman debajo.

3. Crear los contenedores en Podman

- Apache
- Nginx

• Mysql

```
Esteban@UBUNTU-CLASE:~$ sudo podman run -d --name cont_apache \
-p 8080:80 \
-v /mnt/apache_vol:/var/www/html:Z \
apache_custom

sudo podman run -d --name cont_mysql \
-e MYSQL_ROOT_PASSWORD=root \
-e MYSQL_DATABASE=empresa \
-v /mnt/mysql_vol:/var/lib/mysql:Z \
mysql_custom

sudo podman run -d --name cont_nginx \
-p 8081:80 \
-v /mnt/nginx_vol:/usr/share/nginx/html:Z \
nginx_custom
Error: creating container storage: the container name "cont_apache" is already in use by 4a305aa01cf05e2cbd7d4a846146eb1d243fc44a1d3674d0247b8e2b1ac017f
3. You have to remove that container to be able to reuse that name: that name is already in use, or use --replace to instruct Podman to do so.
0ae52b5c8f5032a69ee4f2a30179a8beddf61a193deab4fb809bc4dffea4b02
88db97a0eb1a45e3b5717cce37643589386000e7b14be98c7f530e09152518e2
Esteban@UBUNTU-CLASE:~$ sudo podman ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
0ae52b5c8f50 docker.io/library/mysql_custom:latest mysqld About a minute ago Up About a minute 0.0.0.0:8081->80/tcp cont_mysql
88db97a0eb1a docker.io/library/nginx_custom:latest nginx -g daemon o... About a minute ago Up About a minute 0.0.0.0:8081->80/tcp cont_nginx
```

Observación: Saco error por qué ya se había creado el contenedor de apache, por ende, dice que el nombre ya está en uso

4. Verificación de funcionamiento: sudo podman ps

```
Esteban@UBUNTU-CLASE:~$ sudo podman ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
0ae52b5c8f50 docker.io/library/mysql_custom:latest mysqld 10 minutes ago Up 10 minutes 0.0.0.0:8081->80/tcp cont_mysql
88db97a0eb1a docker.io/library/nginx_custom:latest nginx -g daemon o... 10 minutes ago Up 10 minutes 0.0.0.0:8080->80/tcp cont_nginx
0b4ff112217c2 docker.io/library/apache_custom:latest apachectl -D FORE... 4 seconds ago Up 4 seconds 0.0.0.0:8080->80/tcp cont_apache
Esteban@UBUNTU-CLASE:~$
```

3. Fase 6: DESPLEGAR NETDATA COMO CONTENEDOR

DOCKER

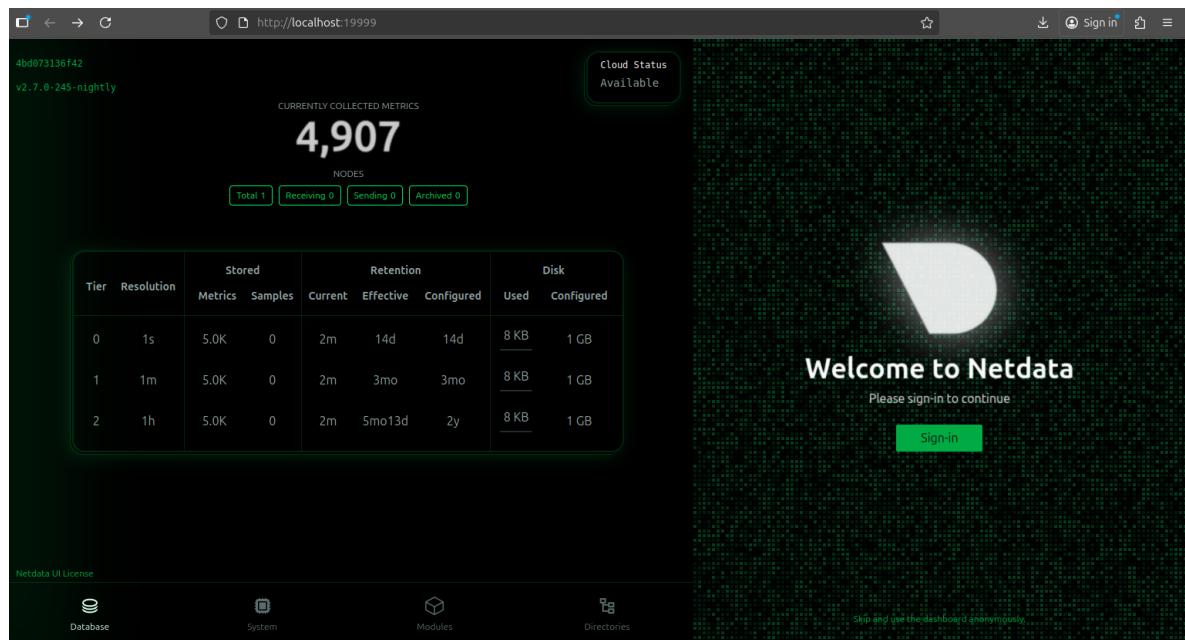
```
Esteban@UBUNTU-CLASE:~$ sudo docker run -d --name netdata \
-p 19999:19999 \
--cap-add SYS_PTRACE \
--security-opt apparmor=unconfined \
-v netdata_lib:/var/lib/netdata \
-v netdata_cache:/var/cache/netdata \
-v /etc/passwd:/host/etc/passwd:ro \
-v /etc/group:/host/etc/group:ro \
-v /proc:/host/proc:ro \
-v /sys:/host/sys:ro \
-v /etc/os-release:/host/etc/os-release:ro \
-v /var/run/docker.sock:/var/run/docker.sock:ro \
netdata/netdata
Unable to find image 'netdata/netdata:latest' locally
latest: Pulling from netdata/netdata
13cc39f8244a: Pull complete
8b167815e432: Pull complete
00ba366457bf: Pull complete
17fcf83258c9: Pull complete
c42266cb4e13: Pull complete
Digest: sha256:6855d3e334864f0b9aca1201256fce97e5952b515f39900a4e1e151c3fc6aeef
Status: Downloaded newer image for netdata/netdata:latest
4bd073136f4283e301354172fdb4d80da9a6b24e79f4e53dc39e16ec395611ab
Esteban@UBUNTU-CLASE:~$ █
```

Observacion: Los commandos que se utilizaron fueron:

- sudo docker run -d --name netdata \
- -p 19999:19999 \
- --cap-add SYS_PTRACE \
- --security-opt apparmor=unconfined \
- -v netdata_lib:/var/lib/netdata \
- -v netdata_cache:/var/cache/netdata \
- -v /etc/passwd:/host/etc/passwd:ro \
- -v /etc/group:/host/etc/group:ro \
- -v /proc:/host/proc:ro \
- -v /sys:/host/sys:ro \
- -v /etc/os-release:/host/etc/os-release:ro \
- -v /var/run/docker.sock:/var/run/docker.sock:ro \
- netdata/netdata

Explicación:

- **netdata/netdata** → imagen oficial
- Se expone **19999** para poder ver el dashboard
- Se montan volúmenes y archivos del host para:
 - Ver procesos
 - Analizar el kernel
 - Detectar contenedores Docker
- docker.sock le permite ver estadísticas de Docker

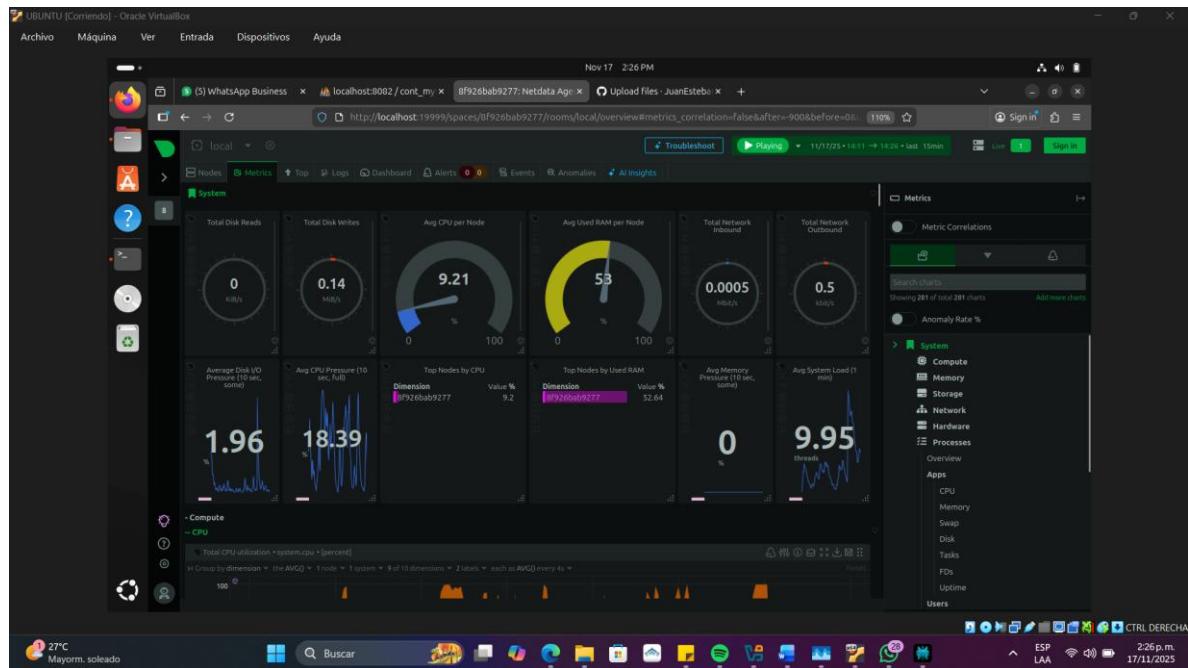


Monitoreo en Tiempo Real con Netdata: Para complementar la infraestructura y validar su rendimiento, se integró Netdata, una herramienta profesional que permite monitorear:

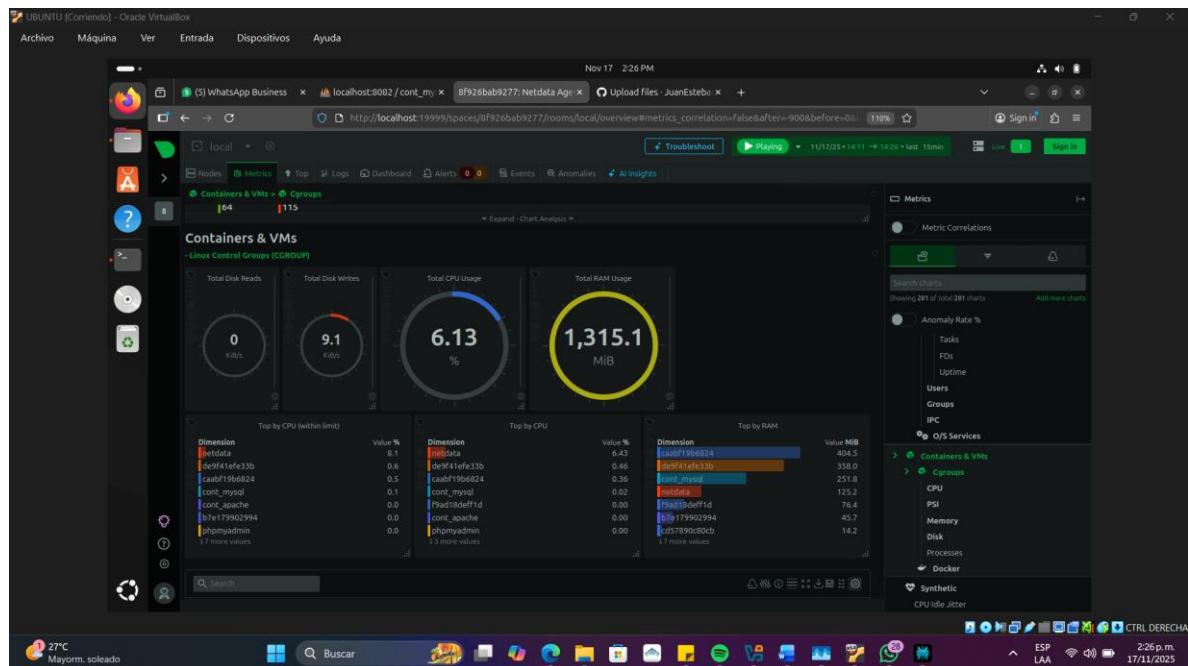
- CPU, RAM y uso de disco
- Estado de los arreglos RAID
- Actividad de lectura/escritura en LVM
- Consumo y métricas de los contenedores (Docker / Podman)
- Métricas específicas de MySQL, Apache y Nginx
- Estado de red y conexiones activas



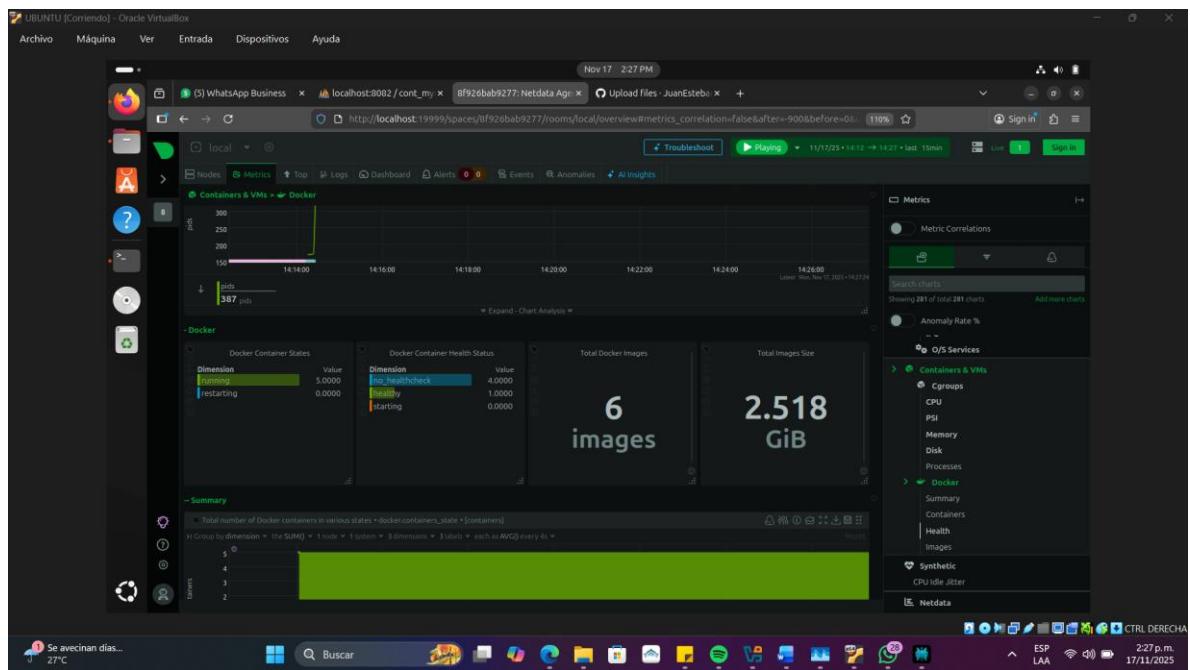
Netdata permite visualizar en tiempo real el comportamiento del entorno y validar el rendimiento general del proyecto.



- Containers identificados:



- Imágenes de los dockers:



4. FASE 7 - IMPLEMENTACIÓN DE NETDATA CON PODMAN

Objetivo: Desplegar Netdata como contenedor utilizando Podman para monitorizar en tiempo real los contenedores y servicios, demostrando la compatibilidad entre Docker y Podman.

- Contexto de la migración a Podman
- Como parte de las buenas prácticas en infraestructura moderna, se decidió migrar el servicio de monitorización Netdata de Docker a Podman debido a las siguientes ventajas técnicas:
- Arquitectura sin daemon: Podman opera sin necesidad de un proceso daemon central, reduciendo la superficie de ataque y mejorando la seguridad.
- Compatibilidad total: Podman puede utilizar las mismas imágenes de Docker y comandos similares.

- Ejecución rootless: Permite ejecutar contenedores sin privilegios de superusuario (aunque para este proyecto se utilizó modo privilegiado para acceso completo al sistema).
- Integración con systemd: Facilita la gestión de contenedores como servicios del sistema

2. Preparación del entorno Podman

Antes de desplegar Netdata, se configuró el socket de Podman para permitir la comunicación con el contenedor de monitorización:

- # Habilitar el socket de Podman: sudo systemctl enable --now podman.socket
 - # Verificar que el socket está disponible: ls -la /run/podman/podman.sock
 - # Configurar permisos del socket: sudo chmod 666 /run/podman/podman.sock
 - Observación: El socket de Podman (/run/podman/podman.sock) es equivalente al de Docker (/var/run/docker.sock) y permite que Netdata acceda a la información de los contenedores en ejecución.

3. Creación de archivos de configuración para Netdata

Para garantizar que Netdata detecte correctamente los contenedores de Podman, se crearon archivos de configuración personalizados en la carpeta del proyecto:

- # Crear estructura de directorios
- mkdir -p ~/ProyectoFinalInfra/ProyectoFinalInfra/netdata/go.d
- cd ~/ProyectoFinalInfra/ProyectoFinalInfra/netdata/go.d

3.1. Configuración del collector de Podman (podman.conf)

Este archivo permite a Netdata conectarse al socket de Podman y recopilar métricas de los contenedores:

jobs:

- name: local
 - url: unix:///host/run/podman/podman.sock
 - collect_container_size: yes
 - timeout: 5

3.2. Configuración del collector de cgroups (cgroups.conf)

Permite monitorizar el consumo de recursos (CPU, memoria, I/O) por cada contenedor:

jobs:

- name: podman-cgroups
 - update_every: 1
 - enable_cgroups: true
 - autodetect: true
 - cgroup_base: "/host/sys/fs/cgroup"

4. Despliegue del contenedor Netdata con Podman

Se ejecutó el siguiente comando para iniciar Netdata con todas las capacidades necesarias para monitorizar el sistema y los contenedores:

```
sudo podman run -d --name netdata \
--hostname="ubuntu-clase" \
--network host \
--pid host \
--privileged \
-e DOCKER_HOST="/host/run/podman/podman.sock" \
-e NETDATA_LISTENER_PORT=19999 \
-v netdata_config:/etc/netdata \
-v netdata_lib:/var/lib/netdata \
-v netdata_cache:/var/cache/netdata \
-v /etc/passwd:/host/etc/passwd:ro \
-v /etc/group:/host/etc/group:ro \
```

```
-v /proc:/host/proc:ro \
-v /sys:/host/sys:ro \
-v /run/podman/podman.sock:/host/run/podman/podman.sock:ro \
--restart unless-stopped \
docker.io/netdata/netdata:latest
```

Parámetro	Función
--network host	Permite a Netdata acceder directamente a la red del host
--pid host	Necesario para ver procesos del sistema y cgroups
--privileged	Da acceso completo al sistema (necesario para monitoreo completo)
-e DOCKER_HOST	Variable de entorno que apunta al socket de Podman
-v /proc:/host/proc:ro	Acceso de solo lectura a información del kernel
-v /sys:/host/sys:ro	Acceso a cgroups para métricas de contenedores
-v /run/podman/podman.sock	Socket para detectar contenedores Podman
--restart unless-stopped	Reinicio automático del contenedor

5. Aplicación de configuraciones personalizadas

Después de que el contenedor inició correctamente, se copiaron las configuraciones creadas anteriormente:

```
# Esperar a que Netdata complete su inicialización
sleep 15
# Copiar archivos de configuración
```

```
sudo podman cp ~/ProyectoFinalInfra/ProyectoFinalInfra/netdata/go.d/podman.conf
netdata:/etc/netdata/go.d/
```

```
sudo podman cp ~/ProyectoFinalInfra/ProyectoFinalInfra/netdata/go.d/cgroups.conf
netdata:/etc/netdata/go.d/
```

Reiniciar Netdata para aplicar cambios

```
sudo podman restart netdata
```

6. Verificación del funcionamiento

6.1. Verificar que el contenedor está corriendo

```
sudo podman ps | grep netdata
```

```
Esteban@UBUNTU-CLASE: $ sudo podman ps | grep netdata
64f655233df3 docker.io/netdata/netdata:latest
13 minutes ago Up 12 minutes
Esteban@UBUNTU-CLASE: ~
```

6.2. Comprobar logs de Netdata

```
sudo podman logs netdata | grep -i "podman\|collector"
```

Observación: Los logs deben mostrar que el collector de Podman se activó correctamente y que detectó los contenedores en ejecución.

6.3 Resultados finales:

Servicio	Tecnología	Puerto	Volumen persistente	Estado de monitoreo
Apache	Podman	8080	/mnt/apache_vol	Monitoreado
MySQL	Podman	3306	/mnt/mysql_vol	Monitoreado
Nginx	Podman	8081	/mnt/nginx_vol	Monitoreado
phpMyAdmin	Podman	8082	N/A	Monitoreado
Netdata	Podman	19999	Volúmenes named	Activo

Conclusiones

El desarrollo del proyecto permitió implementar una infraestructura modular, segura y persistente aplicando principios de redundancia y virtualización. La arquitectura implementada integró múltiples capas de tecnología que, en conjunto, demostraron la viabilidad de sistemas empresariales modernos basados en contenedores.

RAID 1 y LVM garantizaron la integridad y disponibilidad de los datos mediante redundancia por espejo y gestión flexible de volúmenes, permitiendo la expansión y mantenimiento del almacenamiento sin interrupciones del servicio. La sincronización automática de los arreglos RAID proporcionó protección ante fallos de hardware, mientras que LVM facilitó la asignación dinámica de espacio a cada servicio.

Docker y Podman demostraron la portabilidad y eficiencia de los servicios mediante contenedorización. La compatibilidad total entre ambas tecnologías validó la estandarización de imágenes OCI (Open Container Initiative), permitiendo migrar servicios entre plataformas sin modificaciones. Podman, al operar sin daemon central, ofreció ventajas adicionales de seguridad y aislamiento de procesos.

Las pruebas de persistencia confirmaron la conservación de la información tras reinicios, validando la robustez del diseño. Los datos almacenados en volúmenes LVM montados en contenedores sobrevivieron a múltiples ciclos de detención y reinicio, demostrando que la separación entre capa de aplicación y capa de datos es fundamental para la continuidad operacional.

Este proyecto demuestra cómo las tecnologías de contenedores y las estrategias de almacenamiento redundante constituyen la base de la infraestructura moderna en entornos DevOps y empresariales, permitiendo escalabilidad horizontal, despliegues rápidos y recuperación ante desastres.

Resultados específicos de la implementación de Netdata

La integración de Netdata con Podman añadió una capa crítica de observabilidad al proyecto, demostrando:

Compatibilidad total: Las imágenes de Docker funcionaron sin modificaciones en Podman, confirmando la portabilidad de contenedores entre diferentes runtimes. Esta interoperabilidad es esencial en entornos heterogéneos donde coexisten múltiples tecnologías de contenedorización.

Monitoreo en tiempo real: Se logró visualizar métricas de CPU, memoria, disco y red de todos los contenedores y del sistema host con actualización cada segundo. El dashboard de Netdata proporcionó visibilidad inmediata del consumo de recursos por servicio, facilitando la detección temprana de anomalías y la optimización del rendimiento.

Detección automática: Con la configuración correcta del socket de Podman y los collectors personalizados (podman.conf y cgroups.conf), Netdata identificó automáticamente los contenedores en ejecución sin intervención manual. Esta capacidad de auto-descubrimiento reduce significativamente el tiempo de implementación en infraestructuras dinámicas.

Persistencia de datos: Los volúmenes LVM mantuvieron la información histórica de Netdata incluso tras reinicios, permitiendo análisis retrospectivos de tendencias y patrones de uso. La retención de métricas históricas es fundamental para la planificación de capacidad y la identificación de cuellos de botella recurrentes.

Seguridad mejorada: El uso de Podman sin daemon redujo la superficie de ataque en comparación con Docker, eliminando un punto único de fallo y permitiendo ejecución de contenedores con menor privilegio. La arquitectura sin daemon también mitigó riesgos de escalación de privilegios y acceso no autorizado al sistema host.

Esta fase complementó la infraestructura creando un sistema de observabilidad profesional que permite validar el rendimiento, detectar cuellos de botella y mantener la salud operacional del entorno. La capacidad de monitorizar en tiempo real los arreglos RAID, volúmenes LVM y contenedores desde una única interfaz unificó la gestión operacional del proyecto.

Aprendizajes clave

La separación de datos y aplicaciones mediante volúmenes persistentes es esencial para la portabilidad y recuperación ante fallos.

La monitorización proactiva mediante herramientas como Netdata permite anticipar problemas antes de que afecten la disponibilidad del servicio.

La documentación exhaustiva y la automatización mediante scripts facilitan la reproducibilidad y el despliegue consistente de infraestructura.

La integración de múltiples tecnologías (RAID, LVM, contenedores, monitorización) requiere comprensión profunda de cada capa y sus interdependencias.

REFERENCIAS BIBLIOGRÁFICAS

- Tecnologías de contenedorización
- Docker Inc. (2024). Docker Documentation: Docker Hub.

<https://docs.docker.com/>

- Podman. (2024). What is Podman? — Podman documentation.

<https://docs.podman.io/en/latest/Introduction.html>

- Red Hat. (2024). Podman: Managing containers and pods.

<https://podman.io/getting-started/>

- Monitorización y observabilidad

- Netdata Inc. (2024). Netdata Documentation: Learn Netdata.

<https://learn.netdata.cloud/>

- Netdata Inc. (2024). Netdata GitHub Repository: The fastest path to AI-powered full stack monitoring. GitHub. <https://github.com/netdata/netdata>

• Netdata Inc. (2024). Real-Time Monitoring For Infrastructure & Apps. <https://netdata.cloud/>

- Netdata Inc. (2024). Docker Container Monitoring.

<https://learn.netdata.cloud/docs/integrations/containers/docker>

- Netdata Inc. (2024). Podman Container Monitoring.

<https://learn.netdata.cloud/docs/integrations/containers/podman>

- Almacenamiento y volúmenes

- The Linux Foundation. (2024). Logical Volume Manager (LVM) HOWTO. <https://tldp.org/HOWTO/LVM-HOWTO/>

- Red Hat. (2024). Configuring and managing logical volumes. Red Hat Enterprise Linux 9 Documentation.
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/configuring_and_managing_logical_volumes/

- RAID y redundancia
- The Linux Documentation Project. (2024). Linux RAID Wiki.
<https://raid.wiki.kernel.org/>

- Red Hat. (2024). Managing RAID. Red Hat Enterprise Linux 9 Documentation. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/managing_storage_devices/managing_raid_managing-storage-devices

- Servicios web y bases de datos
- The Apache Software Foundation. (2024). Apache HTTP Server Documentation Version 2.4. <https://httpd.apache.org/docs/2.4/>

- NGINX Inc. (2024). NGINX Documentation.
<https://nginx.org/en/docs/>

- Oracle Corporation. (2024). MySQL 8.0 Reference Manual.
<https://dev.mysql.com/doc/refman/8.0/en/>

- phpMyAdmin Contributors. (2024). phpMyAdmin Documentation.

<https://docs.phpmyadmin.net/>

- Virtualización y sistema operativo

- Canonical Ltd. (2024). Ubuntu Server Documentation.

<https://ubuntu.com/server/docs>

- Oracle Corporation. (2024). Oracle VM VirtualBox User Manual.

<https://www.virtualbox.org/manual/>

- Buenas prácticas y arquitectura

- The Linux Foundation. (2024). Linux System Administration

Handbook (5th ed.). Addison-Wesley Professional.

- Docker Inc. (2024). Best practices for writing Dockerfiles.

https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

- Red Hat. (2024). Building, running, and managing containers.

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/building_running_and_managing_containers/

- Referencias académicas

- Turnbull, J. (2014). The Docker Book: Containerization is the new virtualization. James Turnbull.

- Matthias, K., & Kane, S. P. (2018). Docker: Up & Running (2nd ed.).

O'Reilly Media.

- Burns, B., Beda, J., & Hightower, K. (2019). *Kubernetes: Up and Running* (2nd ed.). O'Reilly Media.

- Soppelsa, F., & Kaewkasi, C. (2016). *Native Docker Clustering with Swarm*. Packt Publishing.

- Comunidad y foros técnicos
- Stack Overflow. (2024). Docker - Stack Overflow.

<https://stackoverflow.com/questions/tagged/docker>

- Stack Overflow. (2024). Podman - Stack Overflow.

<https://stackoverflow.com/questions/tagged/podman>

- Netdata Community Forums. (2024). Netdata Community.

<https://community.netdata.cloud/>

- GitHub Community. (2024). Netdata Issues and Discussions.

<https://github.com/netdata/netdata/issues>



UNIQUINDÍO, en conexión territorial

Carrera 15 Calle 12 Norte Tel: (606) 7 35 93 00 Armenia - Quindío - Colombia