

Proyecto Final - Infraestructura Computacional

Diana Maria Garcia Alzate

Código: 1127341120

Daniel Valencia Arenas

Código: 1091202998

Stefania Herrera Ochoa

Código: 1090274506



Profesor: Maycol Cardenas & Jorge Cadavid

Universidad del Quindío

Facultad de Ingeniería

Infraestructura Computacional

Ingeniería de Sistemas y Computación

Grupo 02 Diurno

Armenia, Quindío, 2025

INTRODUCCIÓN

En la actualidad, las organizaciones requieren infraestructuras tecnológicas capaces de ofrecer mayor estabilidad, seguridad, facilidad de administración y escalabilidad. Para lograrlo, el uso de tecnologías como la virtualización y los contenedores es en una práctica ampliamente adoptada, ya que permiten optimizar recursos y desplegar servicios de forma eficiente en un mismo servidor. En este proyecto se construye una infraestructura completa basada en Docker, utilizando diferentes herramientas y comandos del sistema para instalar el entorno, gestionar el almacenamiento y desplegar servicios reales dentro de contenedores.

Para ello, primero se prepara el entorno de almacenamiento utilizando arreglos RAID 1 y administración mediante LVM, garantizando redundancia, flexibilidad y persistencia de datos independiente de los contenedores. Luego se instala Docker mediante herramientas oficiales y se configuran los permisos del sistema para permitir su uso continuo. Con la plataforma lista, se despliegan servicios como Apache, MySQL y Nginx, cada uno asociado a su propio volumen lógico, asegurando que la información generada permanezca incluso tras reinicios o recreación de los contenedores. Finalmente, se crean imágenes personalizadas mediante Dockerfile y se realizan pruebas de funcionamiento y persistencia, validando que la infraestructura responde correctamente a los objetivos del proyecto.

En conjunto, esta implementación demuestra cómo la combinación de RAID, LVM y Docker ofrece una infraestructura moderna, modular y estable, adecuada para entornos donde se requieren servicios confiables y una administración eficiente de los recursos del servidor

FASE 1: creación de RAIDS

Listado de Discos disponibles

El primer paso consistió en identificar los discos disponibles en el sistema. Utilizando el comando `lsblk`, se verificó la presencia de nueve discos duros de 1.5 GB cada uno, identificados desde `sdb` hasta `sdj`

```
userp@ProyectoInfra:~/Desktop$ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0       7:0      0   73.9M  1 loop /snap/core22/2045
loop1       7:1      0   11.1M  1 loop /snap/firmware-updater/167
loop2       7:2      0     4K   1 loop /snap/bare/5
loop3       7:3      0  245.1M  1 loop /snap/firefox/6565
loop4       7:4      0    516M  1 loop /snap/gnome-42-2204/202
loop5       7:5      0   91.7M  1 loop /snap/gtk-common-themes/1535
loop6       7:6      0   10.8M  1 loop /snap/snap-store/1270
loop7       7:7      0   49.3M  1 loop /snap/snapd/24792
loop8       7:8      0    576K  1 loop /snap/snapd-desktop-integration/315
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    1.5G   0 disk
├─md0       9:0      0    1.5G   0 raid1
sdc         8:32     0    1.5G   0 disk
├─md0       9:0      0    1.5G   0 raid1
sdd         8:48     0    1.5G   0 disk
├─md0       9:0      0    1.5G   0 raid1
sde         8:64     0    1.5G   0 disk
sdf         8:80     0    1.5G   0 disk
sdg         8:96     0    1.5G   0 disk
sdh         8:112    0    1.5G   0 disk
sdi         8:128    0    1.5G   0 disk
sdj         8:144    0    1.5G   0 disk
sr0        11:0     1  1024M   0 rom
userp@ProyectoInfra:~/Desktop$
```

Resultado observado: En la imagen se puede apreciar la salida del comando `lsblk`, donde se listan todos los dispositivos de bloque del sistema. Los discos disponibles para la configuración de RAID son:

- `sdb`, `sdc`, `sdd`: 1.5 GB cada uno, destinados para RAID 1 (`md0`)
- `sde`, `sdf`, `sdg`: 1.5 GB cada uno, destinados para RAID 2 (`md1`)
- `sdh`, `sdi`, `sdj`: 1.5 GB cada uno, destinado para RAID 3 (`md2`)

Esta configuración permite crear tres arreglos RAID independientes, cada uno con tres discos en configuración de espejo, garantizando redundancia completa de datos.

Creación de arreglos RAID 1

Se procedió a crear tres arreglos RAID nivel 1 utilizando la herramienta mdadm. Cada RAID fue configurado con tres discos físicos

```
userp@ProyectoInfra:~/Desktop$ sudo mdadm --create --verbose /dev/md0 --level=1 --raid-devices=3 /dev/sdb /dev/sdc /dev/sdd
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 1570816K
Continue creating array?
Continue creating array? (y/n) y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
userp@ProyectoInfra:~/Desktop$
```

- --create /dev/md0 : Crea un nuevo dispositivos RAID con el nombre md0
- --level=1 : Especifica RAID nivel 1 (espejo)
- --raid-devices=3 : Indica que se utilizaran 3 discos en el arreglo
- /dev/sdb /dev/sdc /dev/sdd : Discos físicos que conformarán el RAID

Al ejecutar este comando, el sistema solicita confirmación para poder proceder con la creación del arreglo. La imagen muestra la advertencia de que los metadatos 1.2 serán creados y el inicio del proceso de sincronización de los discos

RAID 2 - md1

Este segundo arreglo RAID sigue la misma configuración que el primero, pero utiliza los discos sde, sdf, sdg. En la imagen se observa el mensaje de confirmación, indicando que el arreglo se creó exitosamente y está operando

```
userp@ProyectoInfra:~/Desktop$ sudo mdadm --create --verbose /dev/md1 --level=1 --raid-devices=3 /dev/sde /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 1570816K
Continue creating array?
Continue creating array? (y/n) y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md1 started.
userp@ProyectoInfra:~/Desktop$
```

RAID 3 - md2

El tercero y último arreglo RAID utiliza los discos restantes (sdh, sdi, sdj). La imagen muestra nuevamente el mensaje de confirmación de creación exitosa del arreglo

```
userp@ProyectoInfra:~/Desktop$ sudo mdadm --create --verbose /dev/md2 --level=1 --raid-devices=3 /dev/sdh /dev/sdi /dev/sdj
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 1570816K
Continue creating array?
Continue creating array? (y/n) y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md2 started.
userp@ProyectoInfra:~/Desktop$
```

Verificación del estado de los RAID

Una vez creados los tres arreglos, se verificó su estado y configuración

```
userp@ProyectoInfra:~/Desktop$ cat /proc/mdstat
Personalities : [raid1]
md2 : active raid1 sdj[2] sdi[1] sdh[0]
      1570816 blocks super 1.2 [3/3] [UUU]

md1 : active raid1 sdg[2] sdf[1] sde[0]
      1570816 blocks super 1.2 [3/3] [UUU]

md0 : active raid1 sdd[2] sdc[1] sdb[0]
      1570816 blocks super 1.2 [3/3] [UUU]

unused devices: <none>
```

La imagen muestra la información detallada de los tres arreglos RAID:

- active raid1 : Indica que los arreglos están activos y en nivel RAID 1
- [3/3] [UUU] : Muestra que los tres discos de cada arreglo están operativos
(U=Up/activo)
- 1534976 blocks : Tamaño efectivo del arreglo (15 GB)
- super 1.2 : Versión de los metadatos RAID utilizados

Creación de volúmenes físicos (PV)

El siguiente paso consistió en preparar cada arreglo RAID como un volumen físico de LVM

```
userp@ProyectoInfra:~/Desktop$ sudo pvcreate /dev/md0
[sudo] password for userp:
Physical volume "/dev/md0" successfully created.
userp@ProyectoInfra:~/Desktop$ sudo pvcreate /dev/md1
Physical volume "/dev/md1" successfully created.
userp@ProyectoInfra:~/Desktop$ sudo pvcreate /dev/md2
Physical volume "/dev/md2" successfully created.
userp@ProyectoInfra:~/Desktop$
```

El comando `pvcreate` inicializa un dispositivo de bloque para su uso con LVM. En la imagen se observa que un mensaje cuando se ejecuta el comando, confirmando que cada arreglo RAID ha sido convertido exitosamente en un volumen físico.

Esta capa de abstracción permite que LVM gestione el espacio de almacenamiento de manera flexible, independientemente de los dispositivos físicos subyacentes

Creación de grupos de volúmenes (VG)

Se crearon tres grupos de volúmenes, uno para cada volumen físico

```
userp@ProyectoInfra:~/Desktop$ sudo vgcreate vg_apache /dev/md0
Volume group "vg_apache" successfully created
userp@ProyectoInfra:~/Desktop$ sudo vgcreate vg_mysql /dev/md1
Volume group "vg_mysql" successfully created
userp@ProyectoInfra:~/Desktop$ sudo vgcreate vg_nginx /dev/md2
Volume group "vg_nginx" successfully created
userp@ProyectoInfra:~/Desktop$
```

Los grupos de volúmenes actúan como conjuntos de almacenamiento desde donde se pueden crear volúmenes lógicos. En la imagen se muestra un mensaje de confirmación, indicando que cada grupo fue creado correctamente.

La nomenclatura utilizada (`vg_apache`, `vg_mysql`, `vg_nginx`) facilita la identificación del propósito de cada grupo y su asociación con el servicio correspondiente.

Creación de volúmenes lógicos (LV)

se creó un volumen lógico dentro de cada grupo de volúmenes

```
userp@ProyectoInfra:~/Desktop$ sudo lvcreate -L 1G -n lv_apache vg_apache
Logical volume "lv_apache" created.
userp@ProyectoInfra:~/Desktop$ sudo lvcreate -L 1G -n lv_mysql vg_mysql
Logical volume "lv_mysql" created.
userp@ProyectoInfra:~/Desktop$ sudo lvcreate -L 1G -n lv_nginx vg_nginx
Logical volume "lv_nginx" created.
userp@ProyectoInfra:~/Desktop$
```

- -L 1G : Especifica el tamaño del volumen lógico (1 GB)
- -n (lv_apache, lv_mysql, lv_nginx) : asigna el nombre al volumen lógico
- vg_apache, vg_mysql, vg_nginx : indica el grupo de volúmenes del cual se tomará el espacio

En la imagen se observa un mensaje, confirmando la creación exitosa. Cada volumen lógico tendrá 1 GB de capacidad, dejando un pequeño margen para metadatos y operaciones futuras.

Verificación de grupos de volúmenes y de volúmenes lógicos

Estas verificaciones confirman que la jerarquía completa de LVM está correctamente configurada y lista para ser utilizada

```

userp@ProyectoInfra:~/Desktop$ sudo pvs
PV          VG          Fmt Attr PSize PFree
/dev/md0    vg_apache  lvm2 a--  <1.50g 508.00m
/dev/md1    vg_mysql   lvm2 a--  <1.50g 508.00m
/dev/md2    vg_nginx   lvm2 a--  <1.50g 508.00m
userp@ProyectoInfra:~/Desktop$ sudo vgs
VG          #PV #LV #SN Attr   VSize VFree
vg_apache   1   1   0 wz--n- <1.50g 508.00m
vg_mysql    1   1   0 wz--n- <1.50g 508.00m
vg_nginx    1   1   0 wz--n- <1.50g 508.00m
userp@ProyectoInfra:~/Desktop$ sudo lvs
LV          VG          Attr   LSize Pool Origin Data%  Meta%   Move Log Cpy%Sync Convert
lv_apache   vg_apache  -wi-a----- 1.00g
lv_mysql    vg_mysql   -wi-a----- 1.00g
lv_nginx    vg_nginx   -wi-a----- 1.00g
userp@ProyectoInfra:~/Desktop$

```

Una vez configurados los volúmenes lógicos se formatean y se monta cada uno en un directorio por cada servicio a utilizar (apache, mysql, nginx).

Formateo de volúmenes lógicos

Cada volumen lógico debe ser formateado con un sistema de archivos antes de poder utilizarse

```

userp@ProyectoInfra:~/Desktop$ sudo mkfs.ext4 /dev/vg_apache/lv_apache
mke2fs 1.47.0 (5-Feb-2023)
Creating filesystem with 262144 4k blocks and 65536 inodes
Filesystem UUID: d3d95e35-bd36-4860-ac3b-64e9b909dc28
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376

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

userp@ProyectoInfra:~/Desktop$ sudo mkfs.ext4 /dev/vg_mysql/lv_mysql
mke2fs 1.47.0 (5-Feb-2023)
Creating filesystem with 262144 4k blocks and 65536 inodes
Filesystem UUID: a55efa55-c1a9-464a-b5f1-fa2425a24984
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376

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

```



```
userp@ProyectoInfra:~/Desktop$ sudo mkfs.ext4 /dev/vg_nginx/lv_nginx
mke2fs 1.47.0 (5-Feb-2023)
Creating filesystem with 262144 4k blocks and 65536 inodes
Filesystem UUID: 1c3d964d-dfaf-43f4-a332-763e427f64d1
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376

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

userp@ProyectoInfra:~/Desktop$
```

El comando `mkfs.ext4` crea un sistema de archivos ext4 (Fourth Extended Filesystem) en cada volumen lógico. Ext4 es el sistema de archivos estándar en Linux, que ofrece:

- Journaling para integridad de datos
- Soporte para archivos y volúmenes grandes
- Mejor rendimiento que sus predecesores
- Herramientas maduras de gestión y recuperación

En la imagen se observa el proceso de creación del sistema de archivos, incluyendo:

- Asignación de inodos
- Escritura de información de superbloque
- Creación de estructuras de journaling
- Confirmación de escritura de superbloques y metadata

Creación de puntos de montaje

Se crearon directorios que servirán como puntos de montaje para cada volumen

```
userp@ProyectoInfra:~/Desktop$ sudo mkdir -p /mnt/apache_data
userp@ProyectoInfra:~/Desktop$ sudo mkdir -p /mnt/mysql_data
userp@ProyectoInfra:~/Desktop$ sudo mkdir -p /mnt/nginx_data
```

El flag `-p` asegura que se creen todos los directorios padres necesarios si no existen. Estos directorios actuarán como la ubicación donde se accederán los datos almacenados en los volúmenes lógicos.

La imagen muestra la verificación donde se observan los tres directorios creados con permisos de root y el timestamp de creación.

Montaje de volúmenes

Se montaron los volúmenes lógicos en sus respectivos puntos de montaje

```
userp@ProyectoInfra:~/Desktop$ sudo mount /dev/vg_apache/lv_apache /mnt/apache_data
userp@ProyectoInfra:~/Desktop$ sudo mount /dev/vg_mysql/lv_mysql /mnt/mysql_data
userp@ProyectoInfra:~/Desktop$ sudo mount /dev/vg_nginx/lv_nginx /mnt/nginx_data
```

El comando mount asocia un dispositivo de almacenamiento con un directorio del sistema de archivos, haciendo que el contenido del dispositivo sea accesible a través de ese directorio.

Verificación del montaje

Se verificó que los volúmenes estén correctamente montados

```
userp@ProyectoInfra:~/Desktop$ df -h | grep mnt
/dev/mapper/vg_apache-lv_apache 974M 24K 907M 1% /mnt/apache_data
/dev/mapper/vg_mysql-lv_mysql 974M 24K 907M 1% /mnt/mysql_data
/dev/mapper/vg_nginx-lv_nginx 974M 24K 907M 1% /mnt/nginx_data
userp@ProyectoInfra:~/Desktop$
```

La imagen muestra la información de todos los sistemas de archivos montados.

Esta confirmación asegura que los volúmenes están accesibles y listos para almacenar datos de los contenedores

FASE 2 : Instalación de Docker

Instalación

Instalación de Docker utilizando el script de la instalación oficial

```
userp@ProyectoInfra:~/Desktop$ curl -fsSL https://get.docker.com -o get-docker.sh
userp@ProyectoInfra:~/Desktop$ sudo sh get-docker.sh
# Executing docker install script, commit: 7d96bd3c5235ab2121bcb855dd7b3f37128ed4
+ sh -c apt-get -qq update >/dev/null
+ sh -c DEBIAN_FRONTEND=noninteractive apt-get -y -qq install ca-certificates curl >/dev/null
sudo+ sh -c install -m 0755 -d /etc/apt/keyrings
+ sh -c curl -fsSL "https://download.docker.com/linux/ubuntu/gpg" -o /etc/apt/keyrings/docker.asc
+ sh -c chmod a+r /etc/apt/keyrings/docker.asc
+ sh -c echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu noble sta
ble" > /etc/apt/sources.list.d/docker.list
+ sh -c apt-get -qq update >/dev/null
+ sh -c DEBIAN_FRONTEND=noninteractive apt-get -y -qq install docker-ce docker-ce-cli containerd.io docker-compose-plugi
n docker-ce-rootless-extras docker-buildx-plugin docker-model-plugin >/dev/null
+ sh -c docker version
```

En el primer comando descarga el script oficial de instalación de Docker y el segundo comando ejecuta el script, configura los repositorios apropiados e instala docker y sus dependencias.

Analisis de salida

En esta imagen se puede observar la información detallada de la instalación de Docker, una parte del mensaje muestra información del cliente y la segunda parte del mensaje muestra la información del servidor

```
Client: Docker Engine - Community
Version:      29.0.0
API version:  1.52
Go version:   go1.25.4
Git commit:   3d4129b
Built:        Mon Nov 10 21:46:31 2025
OS/Arch:      linux/amd64
Context:      default

Server: Docker Engine - Community
Engine:
Version:      29.0.0
API version:  1.52 (minimum version 1.44)
Go version:   go1.25.4
Git commit:   d105562
Built:        Mon Nov 10 21:46:31 2025
OS/Arch:      linux/amd64
Experimental: false
containerd:
Version:      v2.1.5
GitCommit:    fcd43222d6b07379a4be9786bda52438f0dd16a1
runc:
Version:      1.3.3
GitCommit:    v1.3.3-0-gd842d771
docker-init:
Version:      0.19.0
GitCommit:    de40ad0
```

```
=====

To run Docker as a non-privileged user, consider setting up the
Docker daemon in rootless mode for your user:

    dockerd-rootless-setuptool.sh install

Visit https://docs.docker.com/go/rootless/ to learn about rootless mode.

To run the Docker daemon as a fully privileged service, but granting non-root
users access, refer to https://docs.docker.com/go/daemon-access/

WARNING: Access to the remote API on a privileged Docker daemon is equivalent
to root access on the host. Refer to the 'Docker daemon attack surface'
documentation for details: https://docs.docker.com/go/attack-surface/

=====

userp@ProyectoInfra:~/Desktop$
```

En esta imagen se muestra un mensaje informativo importante de Docker que aparece tras la configuración. Este mensaje es importante porque informa que cualquier usuario con acceso al daemon de Docker tiene efectivamente permisos de root en el sistema

Configuración de permisos de usuario

Se agrego el usuario actual al grupo docker para permitir la ejecución de comandos sin privilegios de root

```
userp@ProyectoInfra:~/Desktop$ sudo usermod -aG docker $USER
```

Este comando añade el usuario actual al grupo docker, permitiendo ejecutar comandos Docker sin necesidad de usar sudo. Esto mejora la experiencia de usuario y es una práctica común en entornos de desarrollo, proporcionando mayor agilidad en la gestión de contenedores sin comprometer la seguridad del sistema.

Activación inmediata de permisos

Para aplicar los cambios de grupo sin necesidad de cerrar sesión

```
userp@ProyectoInfra:~/Desktop$ newgrp docker
userp@ProyectoInfra:~/Desktop$
```

Esta imagen muestra la ejecución del comando newgrp docker, que crea una nueva sesión de shell con el grupo docker activado, este comando permite:

- Aplicar inmediatamente los cambios de permisos sin cerrar sesión
- Crear un nuevo shell donde el usuario ya es miembro del grupo docker
- Continuar trabajando sin interrupciones en el flujo de trabajo
- Evitar la necesidad de reiniciar la sesión o el sistema

FASE 3: Creación de contenedores

Contenedor Apache

Creación del contenedor apache

```
userp@ProyectoInfra:~/Desktop$ docker run -d -p 8080:80 -v /mnt/apache_data:/usr/local/apache2/htdocs/ httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
233dec01418c: Pull complete
d7ecded7702a: Pull complete
fab98c44430d: Pull complete
4f4fb700ef54: Pull complete
13c22d886563: Pull complete
4870f70a8556: Pull complete
Digest: sha256:ecfd5ca1bfe1fc5e44a5836c5188bde7f397b50c7a5bb603a017543e29948a01
Status: Downloaded newer image for httpd:latest
7f1703613d33286ae589f9cb29ceb9ab443f932857488497ebc0c95757a98625
```

La imagen muestra el identificador único del contenedor generado tras su creación exitosa, confirmando que el contenedor se está ejecutando en segundo plano

Creación de contenido para Apache

Se creó un archivo .HTML de prueba en el volumen montado

```
userp@ProyectoInfra:/mnt/apache_data$ sudo nano index.html
userp@ProyectoInfra:/mnt/apache_data$
```

En la imagen se muestra la creación del archivo index.html utilizando el editor nano. Este archivo se crea directamente en el volumen LVM montado, esto garantiza persistencia de datos, accesibilidad inmediata, edición externa y protección por redundancia

```
userp@ProyectoInfra: /mnt/apache_data
GNU nano 7.2 index.html
index para apache
```

En esta imagen se muestra el contenido index.html creado en el editor nano, este contenido drive como prueba de concepto para validar que el servidor Apache está sirviendo correctamente archivos desde el volumen persistente

Prueba del servidor apache

Se verificó el funcionamiento del servidor Apache mediando una solicitud HTTP

```
userp@ProyectoInfra:/mnt/apache_data$ curl http://localhost:8080
index para apache
userp@ProyectoInfra:/mnt/apache_data$
```

La imagen muestra la respuesta HTTP completa del servidor Apache, devolviendo el código fuente HTML del archivo index.html creado anteriormente

Contenedor MySQL

Creación del contenedor MySQL

```
userp@ProyectoInfra:~/Desktop$ docker run -d -p 8081:80 -v /mnt/mysql_data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=mysql -
d mysql
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
21aa606d8d58: Pull complete
023a182c62a0: Pull complete
f5f78fcd9ccb: Pull complete
494c372d15c3: Pull complete
dcee80f7340c: Pull complete
480d01bd7a6a: Pull complete
834e15e3ed24: Pull complete
c276de9b5571: Pull complete
0cd145fbb449: Pull complete
5a3f7744d0e7: Pull complete
Digest: sha256:569c4128dfa625ac2ac62cdd8af588a3a6a60a049d1a8d8f0fac95880ecdbbe5
Status: Downloaded newer image for mysql:latest
747396471a697c61e2b235d10c89c1b8ce09f1d93f8955e071962feb83313ee9
userp@ProyectoInfra:~/Desktop$
```

La imagen muestra la confirmación de que MYSQL se ha inicializado correctamente con todas las configuraciones especificadas

Prueba del servidor MySQL

Se verificó el funcionamiento del contenedor MySQL conectándose

```
userp@ProyectoInfra:~/Desktop$ docker exec 747396471a69 mysql -u root -pmysql -e "SHOW DATABASES;"
mysql: [Warning] Using a password on the command line interface can be insecure.
Database
information_schema
mysql
performance_schema
sys
userp@ProyectoInfra:~/Desktop$
```

En la imagen se puede observar comandos SQL ejecutados en la sesión, como la verificación de bases de datos disponibles

Creación del contenedor Nginx

```
userp@ProyectoInfra:/mnt/mysql_data$ docker run -d --name nginx -p 8082:80 -v /mnt/nginx_data:/usr/share/nginx/html nginx:latest
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
de57a609c9d5: Pull complete
0e4bc2bd6656: Pull complete
b5feb73171bf: Pull complete
108ab8292820: Pull complete
53d743880af4: Pull complete
77fa2eb06317: Pull complete
192e2451f875: Pull complete
Digest: sha256:553f64aecdc31b5bf944521731cd70e35da4faed96b2b7548a3d8e2598c52a42
Status: Downloaded newer image for nginx:latest
49097e218d12d3146cb898d9499fd53906e5fb5f2132d88d26251eea38e7a1ad
userp@ProyectoInfra:/mnt/mysql_data$
```

La imagen muestra una confirmación de que el contenedor está corriendo en segundo plano y listo para servir contenido web

Prueba del servidor Nginx

Se verificó el funcionamiento del servidor Nginx mediante una solicitud http

```
userp@ProyectoInfra:~/Desktop$ curl http://localhost:8082
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx/1.29.3</center>
</body>
</html>
userp@ProyectoInfra:~/Desktop$
```

La imagen muestra la respuesta HTTP del servidor Nginx

Verificación de contenedores en ejecución

Se verificó el estado de todos los contenedores creados

```
userp@ProyectoInfra:~/Desktop$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
NAMES
49097e218d12   nginx:latest   "/docker-entrypoint..." About a minute ago Up About a minute 0.0.0.0:8082->80/tcp, [::]:8082->80/tcp
747396471a69   mysql          "docker-entrypoint.s..." 3 days ago    Up 3 days    3306/tcp, 33060/tcp, 0.0.0.0:8081->80/tcp, [::]:8081->80/tcp
7f1703613d33   httpd          "httpd-foreground"       3 days ago    Up 3 days    0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
jovial_cartwright
```

La imagen muestra la tabla completa de contenedores en ejecución. Esta salida confirma que los tres contenedores se están ejecutando simultáneamente sin conflictos, están utilizando sus respectivos volúmenes LVM persistentes, están en estado saludable y operativo

FASE 4: Imágenes personalizadas

Creación de estructura de directorios para imágenes personalizadas

Se creó una estructura de directorio para organizar los archivos de construcción de imágenes

```
userp@ProyectoInfra:~/Desktop$ mkdir -p ~/proyectofinal/{apache,mysql,nginx}
userp@ProyectoInfra:~/Desktop$
```

La imagen muestra la ejecución del comando y la verificación de la estructura creada, organizando de manera clara los componentes de cada imagen

Creación del Dockerfile para Apache

Se creó el Dockerfile para personalizar la imagen de apache

```
userp@ProyectoInfra:~/proyectofinal/apache$ nano Dockerfile
```

```
GNU nano 7.2 Dockerfile *
FROM httpd:2.4
COPY ./public-html/ /usr/local/apache2/htdocs/
EXPOSE 80
CMD ["httpd-foreground"]
```

La imagen muestra el contenido de Dockerfile para apache

Contenido personalizado para Apache

Se creó el subdirectorio que contendrá los archivos HTML y recursos web personalizados para Apache

```
userp@ProyectoInfra:~/proyectofinal/apache$ mkdir -p ~/proyectofinal/apache/public-html
```

La imagen muestra la ejecución del comando sin mensajes de error, confirmando que:

- El directorio public-html se creó exitosamente
- La estructura está lista para recibir archivos HTML y otros recursos web

- El contexto de construcción de Docker está correctamente preparado

Creación del contenido HTML personalizado para Apache

Se creó un archivo HTML personalizado con información específica del proyecto que será servido por Apache

```
userp@ProyectoInfra:~/Desktop$ cat > ~/proyectofinal/apache/public-html/index.html << 'EOF'
> <html>
> <head><title>Apache Personalizado</title></head>
> <body>
> <h1>Servidor Apache</h1>
> <p><strong>Características:</strong></p>
> <ul>
>   <li>Contenedor Docker personalizado</li>
>   <li>Almacenamiento: RAID 1 (3 discos) + LVM</li>
>   <li>Volumen;¿: /dev/vg_apache/lv_apache</li>
>   <li>Persistencia de datos garantizada</li>
> </ul>
> <p><em>Proyecto Final - Virtualización con Docker</em></p>
> </body>
> </html>
> EOF
userp@ProyectoInfra:~/Desktop$
```

El archivo ha sido creado correctamente y está listo para ser incluido en la imagen Docker cuando se ejecute el comando docker build

Creación del Dockerfile para MySQL

Se creó el Dockerfile que define la construcción de la imagen personalizada de MySQL con configuraciones específicas

```
userp@ProyectoInfra:~/proyectofinal/mysql$ cat > ~/proyectofinal/mysql/Dockerfile << 'EOF'
FROM mysql:8.0
ENV MYSQL_ROOT_PASSWORD=dsdinfra
ENV MYSQL_DATABASE=proyectofinal
ENV MYSQL_USER=userp
ENV MYSQL_PASSWORD=infra

EXPOSE 81

COPY ./init-scripts/ /docker-entrypoint-initdb.d/

HEALTHCHECK --interval=30s --timeout=10s --start-period=5s --retries=3 \
  CMD mysql -u root -pdsdinfra -e "SELECT 1;"
EOF
userp@ProyectoInfra:~/proyectofinal/mysql$
```

Creación del script SQL de inicialización para MySQL

Se creó un script SQL que se ejecutará automáticamente cuando el contenedor MySQL se inicialice por primera vez

```
GNU nano 7.2                                01create-tables.sql
CREATE DATABASE IF NOT EXISTS proyectofinal;
USE proyectofinal;
CREATE TABLE IF NOT EXISTS servicios (
  id INT AUTO:INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  estado VARCHAR(20) DEFAULT 'activo',
  fechaCreacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
INSERT INTO servicios (nombre) VALUES
('Apache'),
('MySQL'),
('Nginx');
```

Propósito del script:

- Se ejecuta solo la primera vez que se crea el contenedor
- Registra los tres servicios implementados en el proyecto
- Permite verificar que los datos persisten en el volumen LVM

- Proporciona datos iniciales para consultas SQL

Creación del Dockerfile para Nginx

Se creó el Dockerfile para la imagen personalizada de Nginx

```
userp@ProyectoInfra: ~/proyectofinal/nginx
GNU nano 7.2 Dockerfile
FROM nginx:latest

COPY ./web-content/ /usr/share/nginx/html/

EXPOSE 82

CMD ["nginx", "-g", "daemon off;"]
```

Creación del contenido HTML personalizado para Nginx

Se creó un archivo HTML personalizado con estilos CSS para el servidor Nginx

```
userp@ProyectoInfra:~/Desktop$ mkdir -p ~/proyectofinal/nginx/web-content
userp@ProyectoInfra:~/Desktop$ cat > ~/proyectofinal/nginx/web-content/index.html << 'EOF'
<!DOCTYPE html>
<html>
<head>
<title>Nginx</title>
<style>
body { font-family: Arial, sans-serif; margin: 40px; }
.container { max-width: 800px; margin: 0 auto; }
.feature { background: #f5f5f5; padding: 15px; margin: 10px 0; border-radius: 5px; }
</style>
</head>
<body>
<div class="container">
<h1>Servidor Nginx</h1>
<div class="feature">
<h3>Configuración del Proyecto:</h3>
<ul>
<li><strong>Contenedor:</strong> Imagen personalizada con Dockerfile </li>
<li><strong>Almacenamiento:</strong> RAID 3 con 3 discos +LVM</li>
<li><strong>Volumen:</strong> /dev/vg:nginx/lv:nginx</li>
<li><strong>Persistencia:</strong> Datos almacenados en volumen LVM</li>
</ul>
</div>
<p><strong>Objetivo:</strong> Implementar solución de virtualización con contenedores</p>
</div>
</body>
</html>
EOF
userp@ProyectoInfra:~/Desktop$
```

Este archivo HTML presenta una página web, este archivo prueba que Nginx sirve correctamente el contenido personalizado, incluyendo información específica de la infraestructura implementada, con estilos CSS que mejoran la apariencia y la legibilidad, y permite verificar que el contenido permanece después de reiniciar el contenedor

Construcción de la imagen personalizada de Apache

Se construyó la imagen Docker personalizada para Apache a partir del Dockerfile creado

```

user@ProjectInfra:~/projectofinal/apache$ cd ~/projectofinal/apache && docker build -t ni-apache .
[+] Building 6.2s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 141B
=> [internal] load metadata for docker.io/library/httpd:2.4
=> [internal] load .dockerignore
=> transferring context: 2B
=> [internal] load build context
=> transferring context: 493B
[1/2] FROM docker.io/library/httpd:2.4@sha256:f9b883f993492552sec727b2b29e7967f1e1a0d843f1e8d4f9fcb2fad3f3
=> resolve docker.io/library/httpd:2.4@sha256:4d0b2056:f9b883f993492552sec727b2b29e7967f1e1a0d843f1e8d4f9fcb2fad3f3
=> sha256:5bd45959f7583ac425d9409a6b5143ec3514980664368633965d847f73ee1366d72 191B / 191B
=> sha256:9cd02717fa7514efcbcd10bdcdf1e9159b08643468633965d847f73ee1366d72 13.43MB / 13.43MB
=> sha256:87a14f0839679cd7f2421f2a7c5631a76811358525289c67318c9d8134fc 2.00MB / 2.00MB
=> sha256:4742a9e996d71d036e208e20e1a67278313f3866a96dbdc6Fe6302739 145B / 145B
=> extracting sha256:4742a9e996d71d036e208e20e1a67278313f3866a96dbdc6Fe6302739
=> extracting sha256:4f4b700ef54461ca9251ae0db9a0dc10c5577484ad75ee0db3e836c1
=> extracting sha256:87a14f0839679cd7f2421f2a7c5631a76811358525289c67318c9d8134fc
=> extracting sha256:9cd02717fa7514efcbcd10bdcdf1e9159b08643468633965d847f73ee1366d72
=> extracting sha256:5bd45959f7583ac425d9409a6b5143ec3514980664368633965d847f73ee1366d72
[2/2] COPY ./public-html/ /usr/local/apache2/htdocs/
=> exporting to image
=> exporting layers
=> exporting manifest sha256:016876481746fbfbf45ccc8a5004b323a537aeb61d479787a49b953f8d46c9
=> exporting config sha256:972451f6a9710930315bd9f656a0b58ac7bcccfdca4a6716bb4ab08673541c8
=> exporting attestation manifest sha256:2e352bad4c6e799453d37137236cd1002a99696a752d06b34ee4a6ac7b
=> exporting manifest list sha256:243bf3ee3ae46bf746957dca13ad3c8d199f5c253021982674dcbe4937438d11b
=> naming to docker.io/library/ni-apache:latest
=> unpacking to docker.io/library/ni-apache:latest
user@ProjectInfra:~/projectofinal/apache$

```

La imagen muestra el proceso completo de construcción de Docker, se observa, que docker descarga múltiples capas de la imagen base desde Docker Hub, se ejecutan los 3 pasos del Dockerfile (FROM, COPY, CMD) y al final aparecen dos mensajes confirmando que la imagen se construyó correctamente y que la imagen fue etiquetada como “mi-apache”

Construcción de la imagen personalizada de MySQL

Se construyó la imagen Docker personalizada para MySQL

[illegible]

La imagen muestra el procesos de construcción de la imagen:

- Docker descarga múltiples capas de la imagen base desde Docker Hub
- Se procesan todas las instrucciones definidas (FROM, ENV, EXPOSE, COPY, HEALTHCHECK, CMD)
- Al final se confirma que la imagen se construyó correctamente y que la imagen fue etiquetada como “mi-mysql”

Construcción de la imagen personalizada de Nginx

Se construyó la imagen Docker personalizada para Nginx

```
userp@ProyectoInfra: ~/proyectofinal/nginx$ cd ~/proyectofinal/nginx && docker build -t mi-nginx .
[+] Building 0.7s (7/7) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile             0.0s
=> => transferring dockerfile: 146B                             0.0s
=> [internal] load metadata for docker.io/library/nginx:latest  0.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 1.05kB                               0.0s
=> [1/2] FROM docker.io/library/nginx:latest@sha256:553f64aecdc31b5bf944521731cd70e35da4faed96b2b7548a3d8e2598c5  0.2s
=> => resolve docker.io/library/nginx:latest@sha256:553f64aecdc31b5bf944521731cd70e35da4faed96b2b7548a3d8e2598c5  0.1s
=> [2/2] COPY ./web-content/ /usr/share/nginx/html/            0.1s
=> exporting to image                                           0.3s
=> => exporting layers                                          0.1s
=> => exporting manifest sha256:d4ad38cc3f768d3f4244022213f8e30d1f0589316690183352b1d34d85271cc8  0.0s
=> => exporting config sha256:c4adc14c9ef0f744d653fcac8e01f5a655260b522d74ef169c81d5e1cedc6cf0  0.0s
=> => exporting attestation manifest sha256:40e1d4049f107b46ecfec09b05d73eb1da09542c5f83efa40ce0f3e0b65be04b  0.0s
=> => exporting manifest list sha256:339a37b08c62c463a6102ba68e43c769896ce4cdc7a90a817cd5efe13f3603cc  0.0s
=> => naming to docker.io/library/mi-nginx:latest              0.0s
=> => unpacking to docker.io/library/mi-nginx:latest            0.0s
userp@ProyectoInfra: ~/proyectofinal/nginx$
```

Esta imagen muestra el proceso de construcción (docker build) de una imagen personalizada

llamada mi-nginx, basada en un Dockerfile

La terminal indica cada etapa del build:

- Carga del contexto
- lectura del Dockerfile
- pasos internos del build
- exportación de capas

al final aparece finished, lo que confirma que la imagen se creó de forma correcta

Listado de imágenes filtrado por nombre

```
userp@ProyectoInfra:~/proyectofinal/nginx$ docker images | grep mi-  
WARNING: This output is designed for human readability. For machine-readable output, please use --format.  
mi-apache:latest      243bef3ea46b      175MB      45.2MB  
mi-mysql:latest       a8ef6f73872d      1.07GB     235MB  
mi-nginx:latest       339a37b08c62      225MB      59.8MB  
userp@ProyectoInfra:~/proyectofinal/nginx$
```

Se muestra el comando `docker images | grep mi-`, el cual filtra las imágenes locales cuyo nombre comienza con `mi-`

El resultado muestra tres imágenes personalizadas:

- `mi_apache`
- `mi_mysql`
- `mi_nginx`

Cada una con su tamaño y su ID de imagen correspondiente

Detección y eliminación de contenedores

```
userp@ProyectoInfra:~/proyectofinal/nginx$ docker stop 7f1703613d33 747396471a69 49097e218d12  
7f1703613d33  
747396471a69  
49097e218d12  
userp@ProyectoInfra:~/proyectofinal/nginx$ docker rm 7f1703613d33 747396471a69 49097e218d12  
7f1703613d33  
747396471a69  
49097e218d12  
userp@ProyectoInfra:~/proyectofinal/nginx$
```

En esta imagen se ven los comandos `docker stop` y `docker rm` aplicados a tres contenedores distintos. Primero se detienen los contenedores activos (tres IDs diferentes) y luego se eliminan definitivamente del sistema. La salida es simplemente la confirmación de los IDs que fueron detenidos y eliminados.

Ejecución y creación del contenedor Apache

```
userp@ProyectoInfra:~/Desktop$ docker run -d --name apache-container -p 8080:80 -v /mnt/apache_data:/usr/local/apache2/htdocs mi-apache
838d25517e10e14e6eda21c64504104421b9bf780870362890e03977eab098cc
```

Aquí se ejecuta el comando docker run, creando un contenedor llamado apache-container basado en la imagen personalizada mi-apache

Ejecución y creación del contenedor MySQL

```
userp@ProyectoInfra:~/proyectofinal/mysql/init-scripts$ docker run -d --name mysql-container -p 8081:80 -v /mnt/mysql_data:/var/lib/mysql -e MYSQL_PASSWORD=dsdinfra -d mi-mysql
0d8903d1124ff2af4f8c2f595cdac47f91c61c8d6719d4133f176551b6514968
```

En esta imagen se ve la ejecución del comando docker run para crear un contenedor llamado mysql-container

```
userp@ProyectoInfra:~/proyectofinal/mysql/init-scripts$ docker run -d --name nginx-container -p 8082:80 -v/mnt/nginx_data:/usr/share/nginx/html mi-nginx
9c744d6ddb91b803a8562b7911598563b986c007ad4478cb6b141755a31c43e4
userp@ProyectoInfra:~/proyectofinal/mysql/init-scripts$
```

```
userp@ProyectoInfra:~/proyectofinal/mysql/init-scripts$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
9c744d6ddb91	mi-nginx	"/docker-entrypoint..."	54 seconds ago	Up 53 seconds	82/tcp, 0.0.0.0:8082->80/tcp, [::]:8082->80/tcp
0d8903d1124f	mi-mysql	"docker-entrypoint.s..."	2 minutes ago	Up 2 minutes (healthy)	81/tcp, 3306/tcp, 33060/tcp, 0.0.0.0:8081->80/tcp, [::]:8081->80/tcp
838d25517e10	mi-apache	"httpd-foreground"	15 minutes ago	Up 15 minutes	0.0.0.0:8080->80/tcp, [::]:8080->80/tcp

```
userp@ProyectoInfra:~/proyectofinal/mysql/init-scripts$
```

FASE 5): Pruebas de persistencia y funcionamiento

PRUEBA 1: Apache- Persistencia de archivos web

Se crea un nuevo archivo de prueba en el volumen

```
userp@ProyectoInfra:~/Desktop$ echo "<h1>Archivo de prueba</h1>" | sudo tee /mnt/apache_data/persistente.html
<h1>Archivo de prueba</h1>
```

Se verifica el acceso al archivo con curl

```
<h1>Archivo de prueba</h1>
userp@ProyectoInfra:~/Desktop$ curl http://localhost:8080/persistente.html
<h1>Archivo de prueba</h1>
userp@ProyectoInfra:~/Desktop$
```

Se reinicia el contenedor y se verifica la persistencia después del reinicio.

```
userp@ProyectoInfra:~/Desktop$ docker restart apache-container
apache-container
userp@ProyectoInfra:~/Desktop$ curl http://localhost:8080/persistente.html
<h1>Archivo de prueba</h1>
userp@ProyectoInfra:~/Desktop$
```

PRUEBA 2: MySQL - Creación y consulta de bases de datos.

Se crea una base de datos y una tabla

```
userp@ProyectoInfra:~/Desktop$ docker exec mysql-container mysql -u root -pdsdinfra -e "
CREATE DATABASE IF NOT EXISTS empleados;
USE empleados;
CREATE TABLE IF NOT EXISTS departamentos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50)
);
INSERT INTO departamentos (id, nombre) VALUES
(NULL, 'Diana'),
(NULL, 'Daniel'),
(NULL, 'Tefa');
"
mysql: [Warning] Using a password on the command line interface can be insecure.
userp@ProyectoInfra:~/Desktop$
```

Verificamos sus datos

```
userp@ProyectoInfra:~/Desktop$ docker exec mysql-container mysql -u root -pdsdinfra -e
> USE empleados;
> SELECT * FROM departamentos;
> "
mysql: [Warning] Using a password on the command line interface can be insecure.
id      nombre  responsable
1       Diana   NULL
2       Daniel  NULL
3       Tefa    NULL
```

Reiniciamos el contenedor y volvemos a comprobar persistencia.

```

userp@ProyectoInfra:~/Desktop$ docker restart mysql-container
mysql-container
userp@ProyectoInfra:~/Desktop$ docker exec mysql-container mysql -u root -pdsdinfra -e "
USE empleados;
SELECT * FROM departamentos;
"
mysql: [Warning] Using a password on the command line interface can be insecure.
id      nombre responsable
1       Diana      NULL
2       Daniel     NULL
3       Tefa       NULL
userp@ProyectoInfra:~/Desktop$

```

PRUEBA 3: Nginx

Se crea un archivo de prueba y se verifica en el contenedor.

```

userp@ProyectoInfra:~/Desktop$ echo "<h1>Archivo prueba Nginx</h1>" | sudo tee /mnt/nginx_data/nginx.html
[sudo] password for userp:
<h1>Archivo prueba Nginx</h1>
userp@ProyectoInfra:~/Desktop$ curl http://localhost:8082/nginx.html
<h1>Archivo prueba Nginx</h1>
userp@ProyectoInfra:~/Desktop$

```

Se reinicia el contenedor y se verifica la persistencia del archivo.

```

userp@ProyectoInfra:~/Desktop$ docker restart nginx-container
nginx-container
userp@ProyectoInfra:~/Desktop$ curl http://localhost:8082/nginx.html
<h1>Archivo prueba Nginx</h1>
userp@ProyectoInfra:~/Desktop$

```

Conclusiones

La implementación demostró que el uso conjunto de RAID, LVM y Docker permite construir una infraestructura estable, segura y modular. Los arreglos RAID garantizaron protección ante fallos, LVM aportó una gestión flexible del almacenamiento y Docker facilitó el despliegue de servicios independientes con imágenes personalizadas. Las pruebas confirmaron que los datos permanecen incluso después de reiniciar los contenedores, validando la persistencia y la confiabilidad del sistema. En conjunto, el proyecto cumple los objetivos propuestos y se ajusta a las necesidades de una infraestructura moderna en entornos reales

