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.

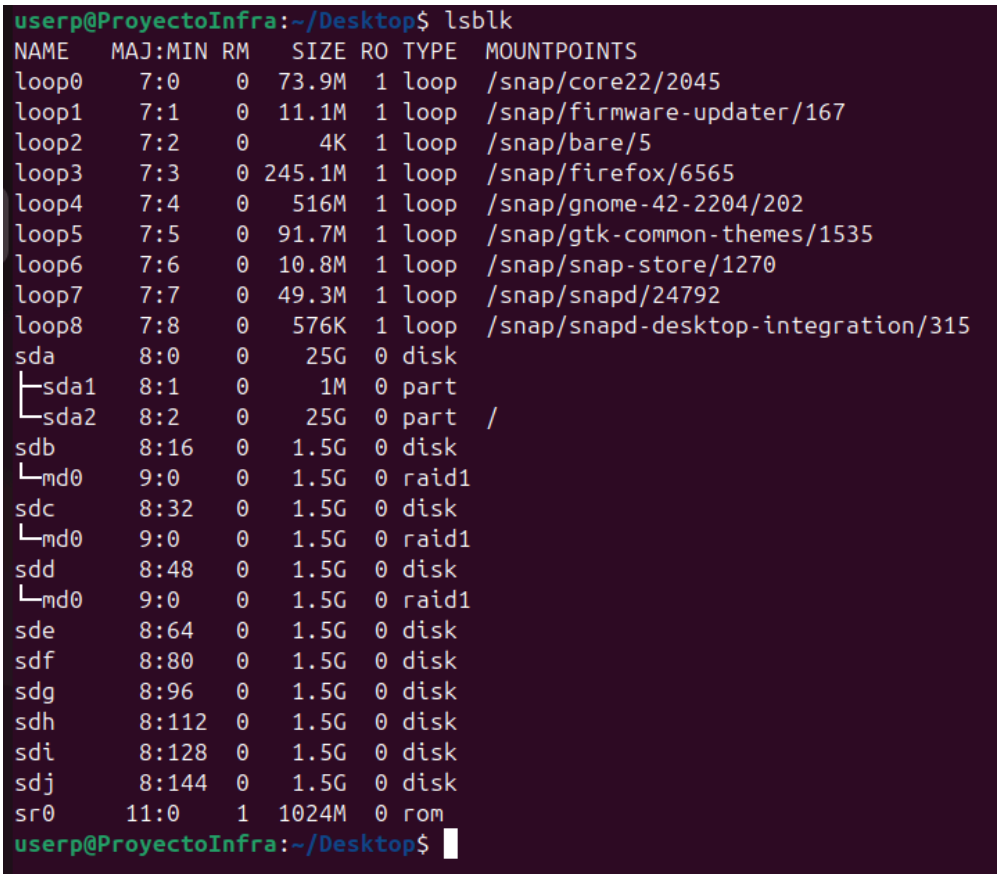
Para ello, primero se prepara el entorno de almacenamiento utilizando arreglos RAID 1 y administración mediante LVM, garantizando redundancia, flexibilidad y persistencia de datos independiente de los contenedores. Luego se instala Docker mediante herramientas oficiales y se configuran los permisos del 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



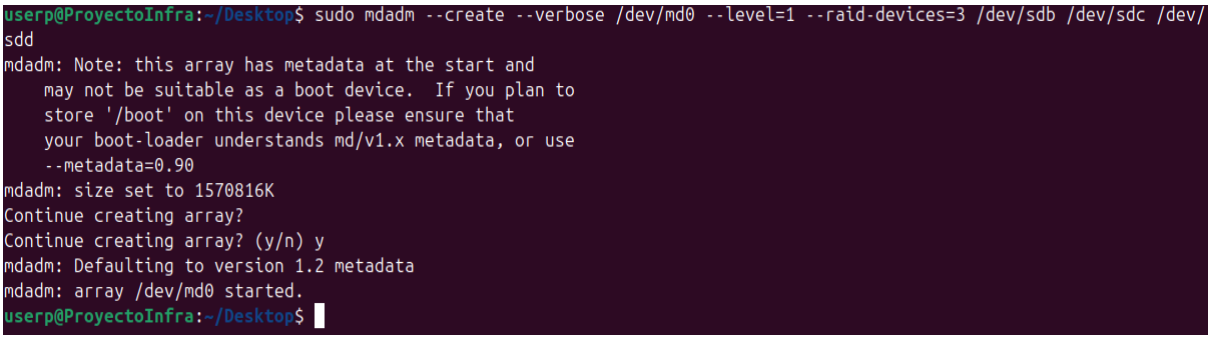
Resultado observado: En la imagen se puede apreciar la salida del comando lsblk, donde se listan todos los dispositivos de bloque del sistema. Los discos disponibles para la configuración de RAID son:

- 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 mdma. Cada RAID fue configurado con tres discos físicos

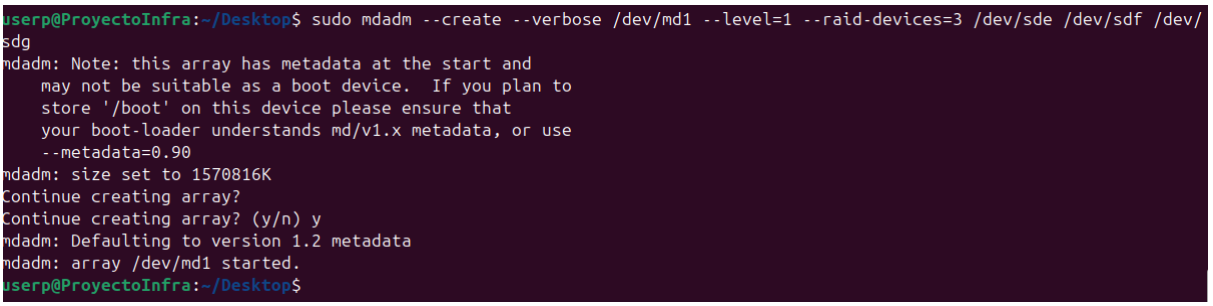


- –create /dev/md0 : Crea un nuevo dispositivos RAID con el nombre md0
- –level=1 : Especifica RAID nivel 1 (espejo)
- –raid-devices=3 : Indica que se utilizaran 3 discos en el arreglo
- /dev/sdb /dev/sdc /dev/sdd : Discos físicos que conformarán el RAID

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



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)

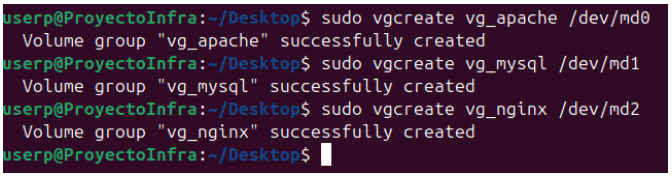
El siguiente paso consistió en preparar cada arreglo RAID como un volumen físico de LVM

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

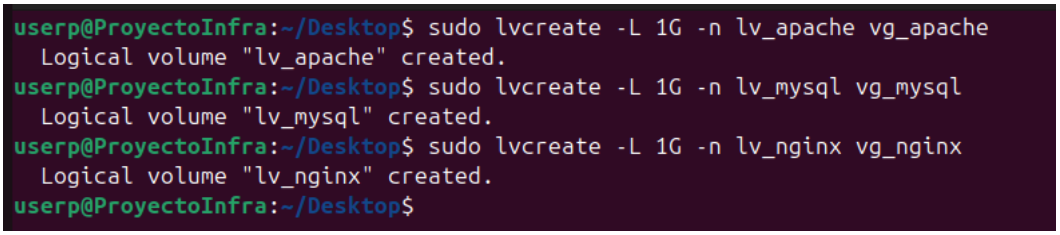


Los grupos de volúmenes actúan como conjuntos de almacenamiento desde donde se pueden crear volúmenes lógicos. En la imagen se muestra un mensaje de confirmación, indicando que cada grupo fue creado correctamente.

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



- -L 1G : Especifica el tamaño del volumen lógico (1 GB)
- -n (lv_apache, lv_mysql, lv_nginx) : asigna el nombre al volumen lógico
- vd_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

```
serp@ProyectoInfra:~/Desktop$ sudo mkfs.ext4 /dev/vg_apache/lv_apache
ke2fs 1.47.0 (5-Feb-2023)
reating filesystem with 262144 4k blocks and 65536 inodes
ilesystem UUID: d3d95e35-bd36-4860-ac3b-64e9b909dc28
uperblock backups stored on blocks:
        32768, 98304, 163840, 229376

llocating group tables: done
riting inode tables: done
reating journal (8192 blocks): done
riting superblocks and filesystem accounting information: done

serp@ProyectoInfra:~/Desktop$ sudo mkfs.ext4 /dev/vg_mysql/lv_mysql
ke2fs 1.47.0 (5-Feb-2023)
reating filesystem with 262144 4k blocks and 65536 inodes
ilesystem UUID: a55efa55-c1a9-464a-b5f1-fa2425a24984
uperblock backups stored on blocks:
        32768, 98304, 163840, 229376

llocating group tables: done
riting inode tables: done
reating journal (8192 blocks): done
riting 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

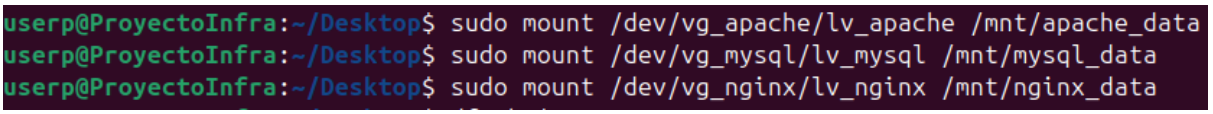
```
userp@ProyectoInfra:~/Desktop$ sudo mkdir -p /mnt/apache_data
userp@ProyectoInfra:~/Desktop$ sudo mkdir -p /mnt/mysql_data
userp@ProyectoInfra:~/Desktop$ sudo mkdir -p /mnt/nginx_data
```

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

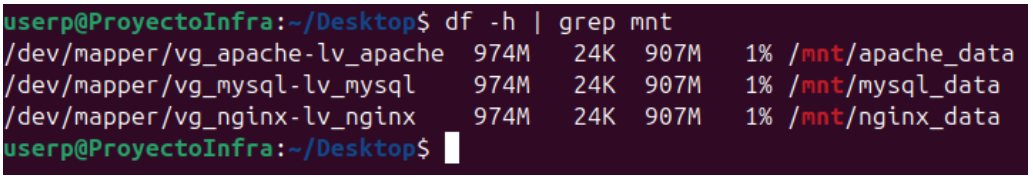
Se montaron los volúmenes lógicos en sus respectivos puntos de montaje



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



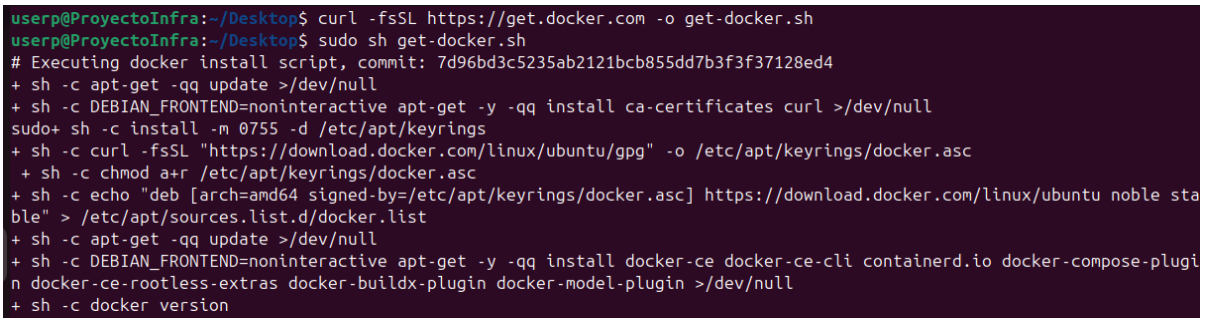
La imagen muestra la información de todos los sistemas de archivos montados.

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



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 sesion o el sistema

FASE 3: Creación de contenedores

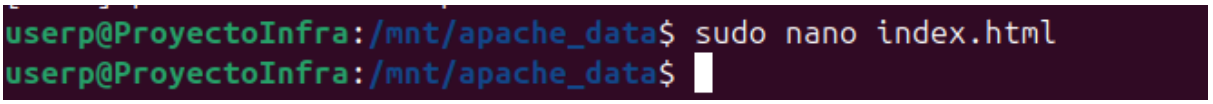
Contenedor Apache

Creación del contenedor apache

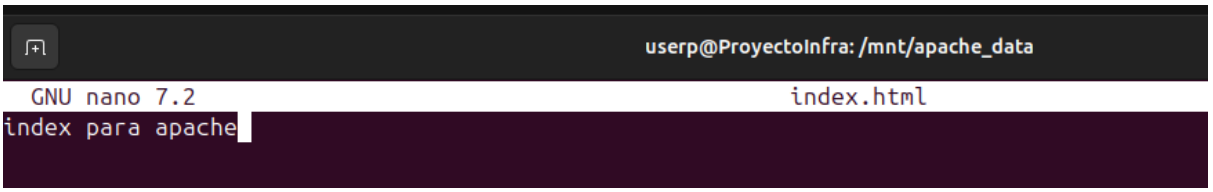
La imagen muestra el identificador único del contenedor generado tras su creación exitosa, confirmando que el contenedor se está ejecutando en segundo plano

Creación de contenido para Apache

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



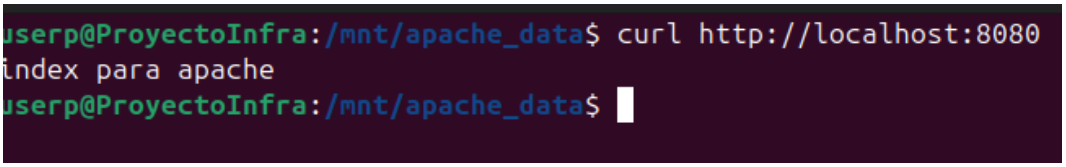
En la imagen se muestra la creación del archivo index.html utilizando el editor nano. Este archivo se crea directamente en el volumen LVM montado, esto garantiza persistencia de datos, accesibilidad inmediata, edición externa y protección por redundancia



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



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

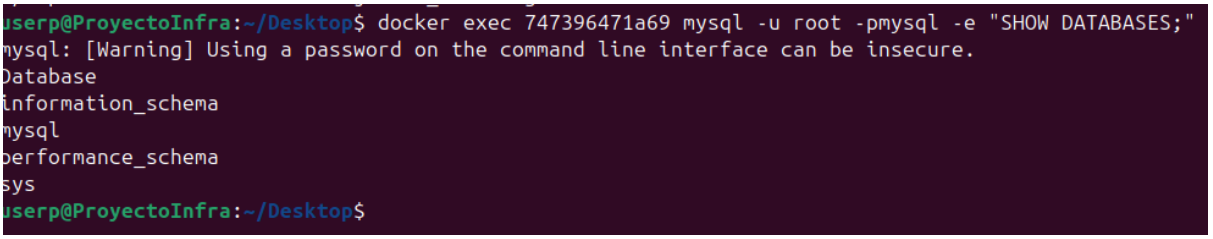
Contenedor MySQL

Creación del contenedor MySQL

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

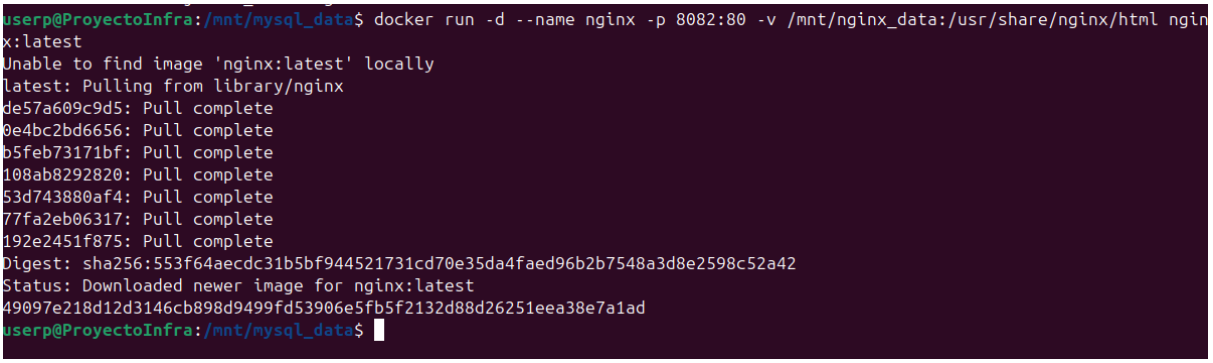
Prueba del servidor MySQL

Se verificó el funcionamiento del contenedor MySQL conectándose



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

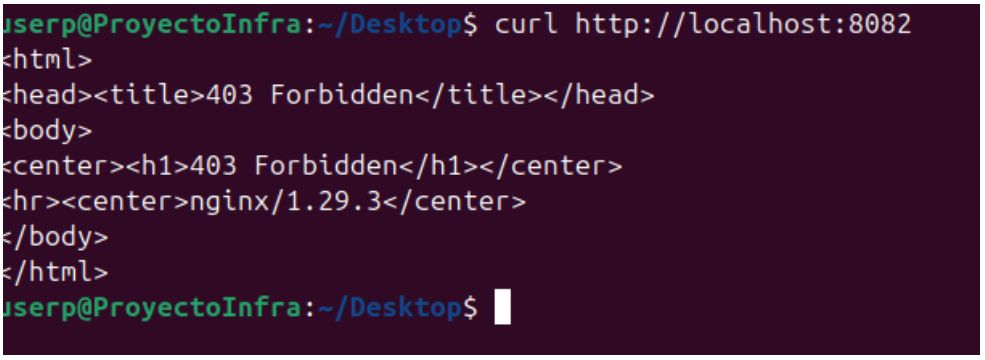
Creación del contenedor Nginx



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



La imagen muestra la respuesta HTTP del servidor Nginx

Verificación de contenedores en ejecución

Se verificó el estado de todos los contenedores creados

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>Caracteristicas:</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

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

Creación del contenido HTMl personalizado para Nginx

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

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

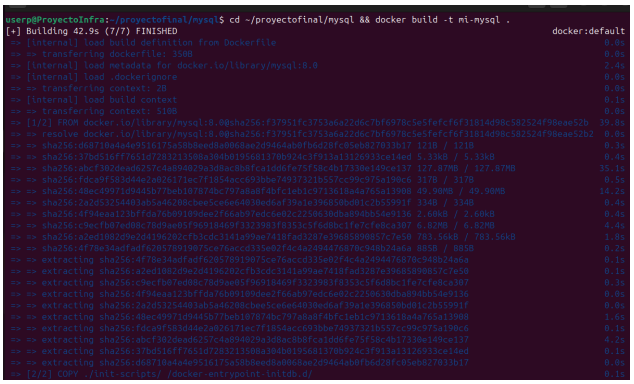
Construcción de la imagen personalizada de Apache

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

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

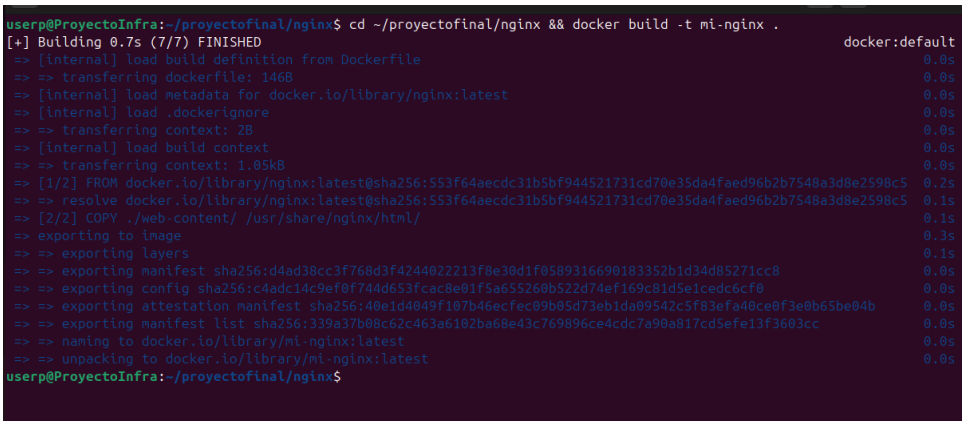


La imagen muestra el procesos de construcción de la imagen:

- 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



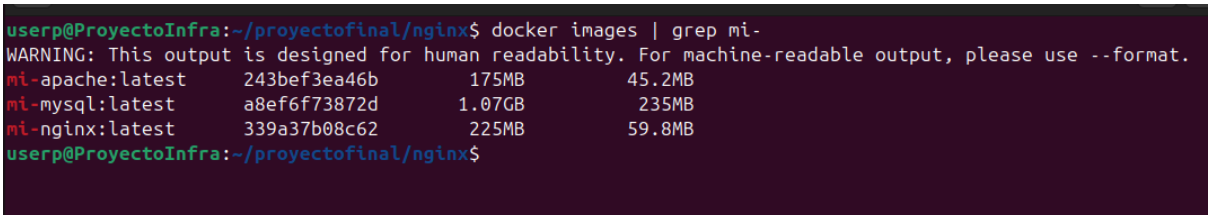
Esta imagen muestra el proceso de construcción (docker build) de una imagen personalizada llamada mi-nginx, basada en un Dockerfile

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



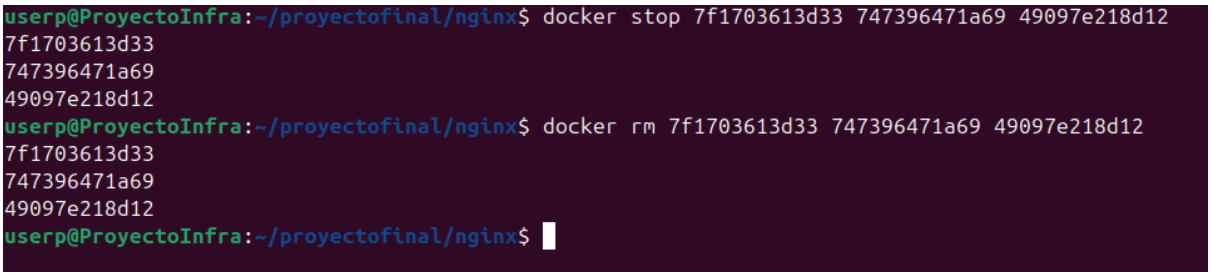
Se muestra el comando docker images | grep mi-, el cual filtra las imágenes locales cuyo nombre comienza con mi-

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



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/h
tdocs 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_da
ta:/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_dat
a:/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
                                   NAMES
9c744d6ddb91   mi-nginx    "/docker-entrypoint.…"   54 seconds ago   Up 53 seconds             82/tcp, 0.0.0.0:8082->80/t
cp, [::]:8082->80/tcp                         nginx-container
0d8903d1124f   mi-mysql    "docker-entrypoint.s…"   2 minutes ago    Up 2 minutes (healthy)    81/tcp, 3306/tcp, 33060/tc
p, 0.0.0.0:8081->80/tcp, [::]:8081->80/tcp    mysql-container
838d25517e10   mi-apache   "httpd-foreground"       15 minutes ago   Up 15 minutes             0.0.0.0:8080->80/tcp, [::]
:8080->80/tcp                                 apache-container
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

```
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  reponsable
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  reponsable
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