

## Prácticas- ISE

### Práctica 1: Instalación y configuración de Ubuntu Server

RAIDs (Redundant Array of Independent Disks) : se encarga de tener varios discos conectados entre sí para evitar pérdida de datos o para acelerar la lectura y escritura. Hay 5 tipos. y otros que son combinaciones de los anteriores.

5 posibles configuraciones estándar:

- RAID 0 : mínimo necesita 2 discos duros, parte el dato que le llega por la mitad, una mitad a un disco y la otra mitad al otro disco de forma que ganamos velocidad de lectura y escritura, porque se hace de forma paralela. Si se rompe un disco duro se pierde todo, por tanto no tiene seguridad.  
Hay otras soluciones que permiten la redundancia.
- RAID1: garantiza redundancia, porque tenemos los dos disco duros. y el mismo dato de antes lo guarda duplicado en ambos discos. Proporciona un poco de seguridad. Si cae un disco tenemos el otro.  
Después del 1 tenemos combinaciones
- RAID5: mínimo 3 discos duros. En el primero, si nos llega un dato se comporta como un RAID0, parte el dato y la mitad a cada disco( en los dos primeros), el tercero guarda de alguna forma información para recuperar el dato perdido, esto se conoce como paridad. Si se cae el tercer disco duro hay problema
- RAID 6: mínimo 4 discos duros. Dos de datos y dos de paridad. De forma que si cae un disco de paridad no pasa nada.
- RAID 10: combina el 0 y el uno. En la parte de arriba tenemos un raid 0 y en los extremos de este enganchamos unos raid 1. El dato llega y se parte en ambos subniveles de raid 1 y en este nivel se redunda. Esto permite mejorar la velocidad mientras nos aseguramos de que tenemos redundancia. Mientras que no caiga un subnivel entero no hay problema.
- Luego tenemos mas combinaciones de todos estos RAIDs

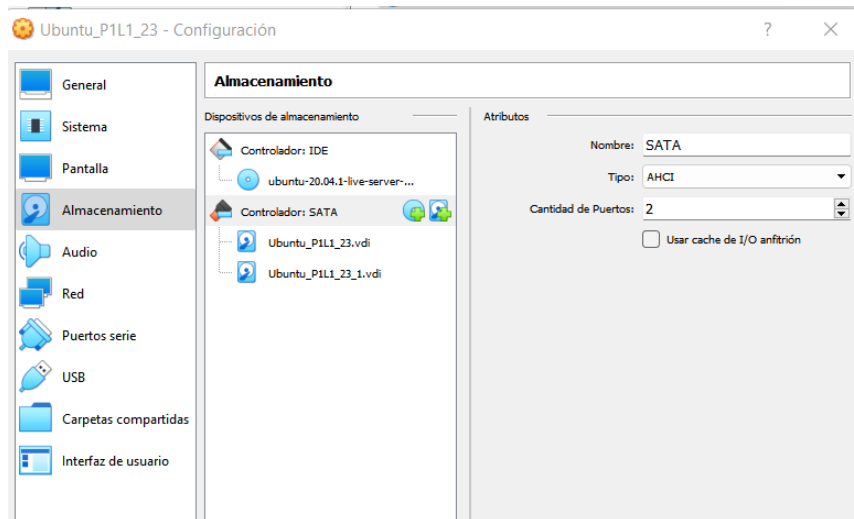
### Diferencias de hacer un raid por software y un raid por hardware

- Por hardware : tenemos una controladora enchufada al servidor y le quitas peso al SO y cpu. Más caro.
- Por software: lo gestiona la cpu, más flexible

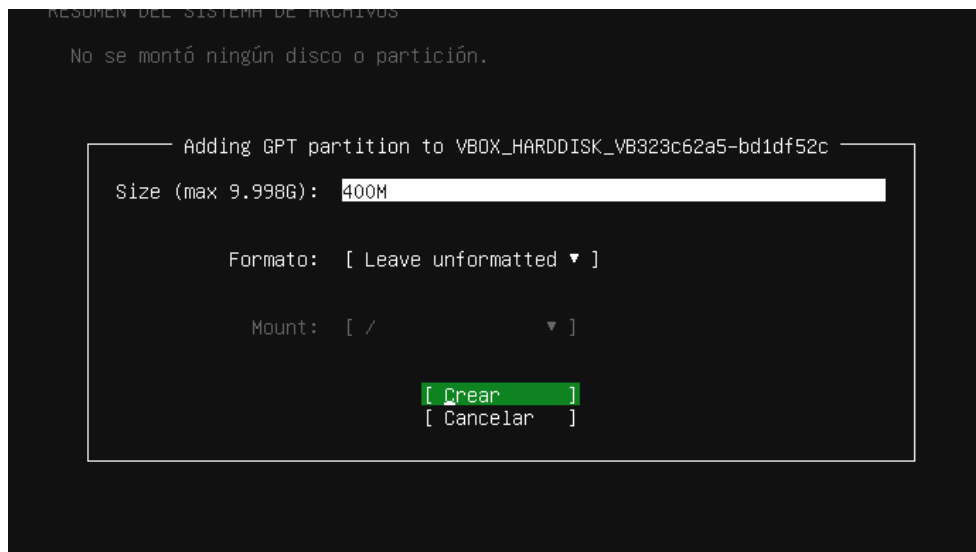
## Lección 1

### Práctica 1:

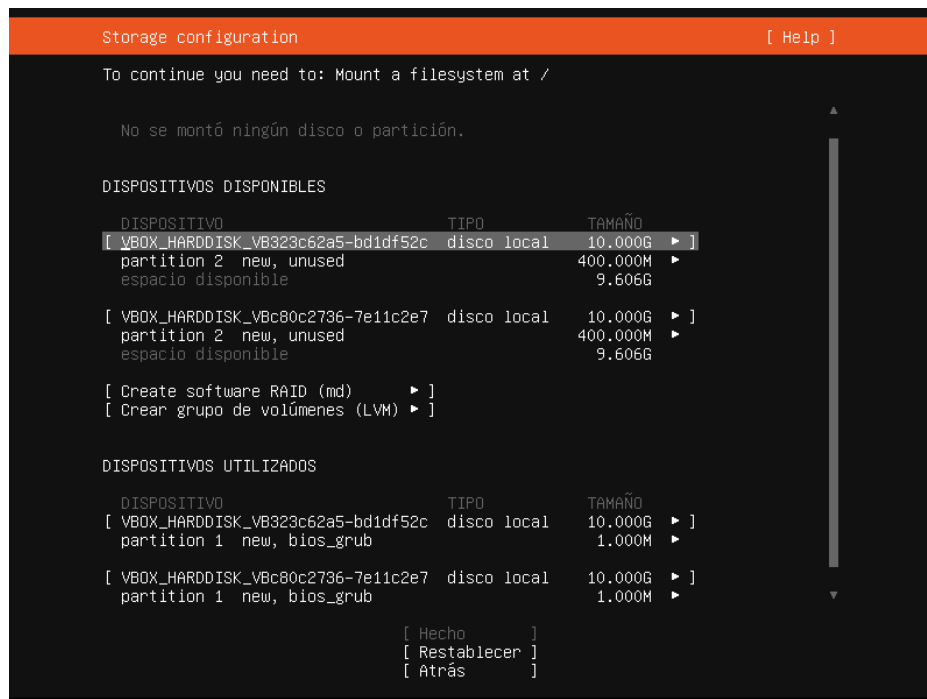
Creamos dos discos duros en la máquina virtual



Creamos la primera partición



En el segundo disco le damos a que nos cree directamente el Boot y vamos a crear la partición.

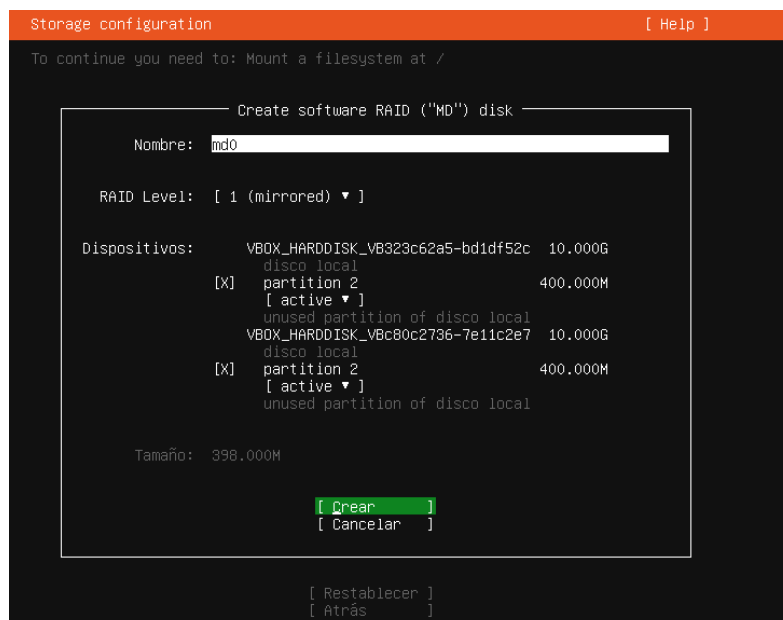


Y por qué los dejamos sin configuración, queremos un RAID1, entonces a raíz de la partición 1 y 2 vamos a crear una primera abstracción para nuestra RAID, denominada **md0**.

Vamos a hacer un RAID software.

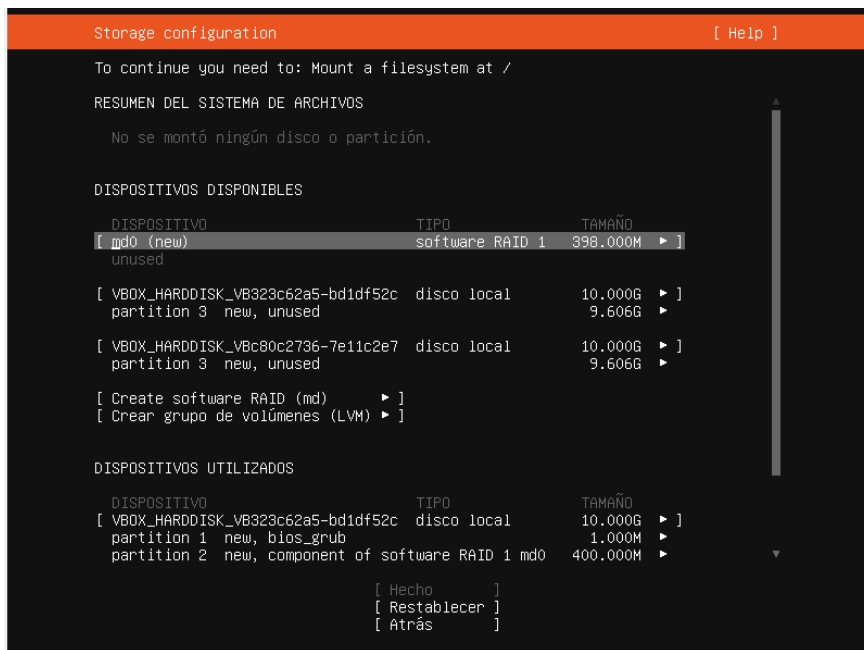
Vamos a crearlo., en **create software RAID(MD)**:

**Seleccionamos el tipo de RAID que queremos y las particiones que vamos a usar para ello.**

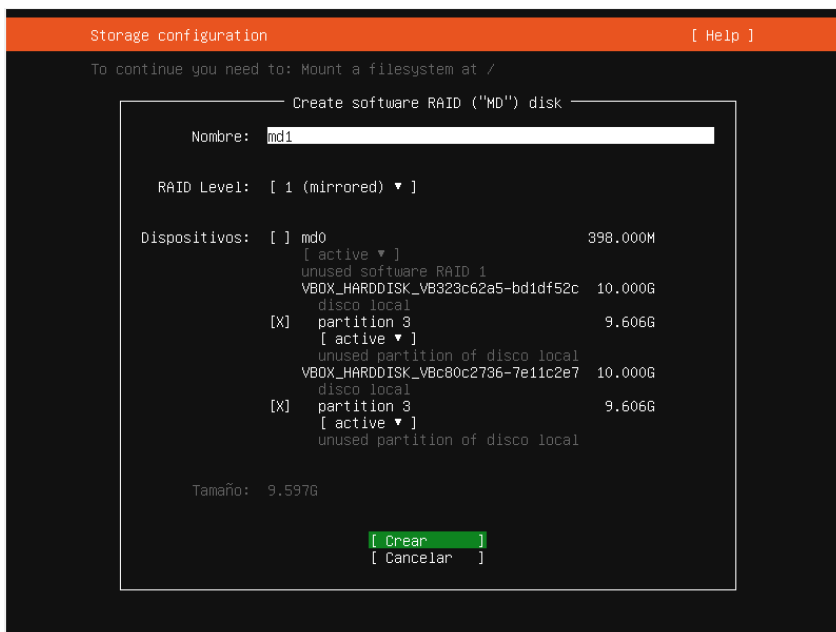


Ahora vamos a crear el RAID 1 para la partición que va a contener el resto de datos.

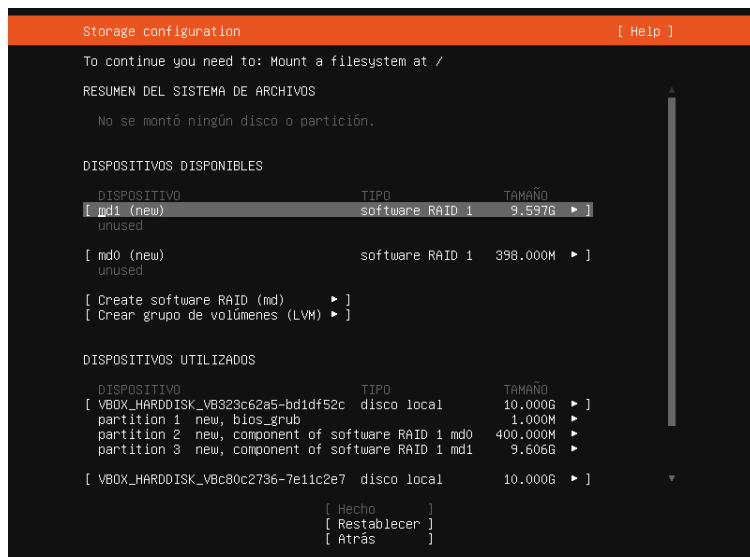
Igual que hemos creado la anterior de 400M, pero sin especificar tamaño para que coja todo el restante



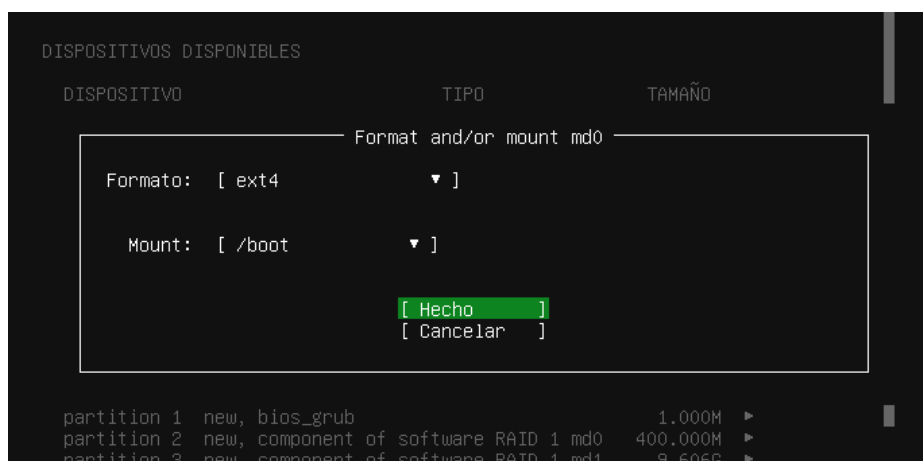
Y ahora vamos a crear la abstracción md1. Creamos nuestro software raid, como el anterior, en este caso md1



Y ya vemos que tenemos nuestras dos abstracciones md0 y md1.



Vamos a asignar al `p` asignarle el punto de montaje y el formato a la partición de arranque.  
`md0` → `format`



**(Tras este paso obtener el esquema pintado en negro de la imagen siguiente)**

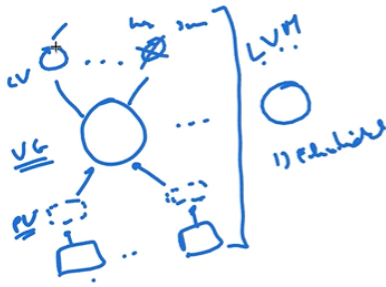
Y ahora vamos a crear la configuración que se nos exige de los volúmenes lógicos.  
 ¿Qué son los volúmenes lógicos? Abstracción lógica de nuestro dispositivo físico, que se va a aglutinar en un volumen group. Ese grupo de volúmenes que agrupa a los físicos, se agrupan a su vez en volúmenes lógicos

Todo esto se conoce como el LVM ( logical volume manager).**(esquema azul de la imagen siguiente)**. Ventajas:

- flexibilidad, a nivel lógico podemos borrar, redimensionar etc
- añadir/quitar volúmenes físicos, sin redefinir el esquema lógico
- snapshot, instantáneas de los grupos lógicos que podemos tomar

LVM (Logical Volumen Manager) : Abstracción lógica de un punto del sistema

VG( Volumen Group) : congregación de volúmenes físicos.



md0 lo hemos definido ya como /boot

**Ahora vamos a crear nuestro grupo de volúmenes (LVM)**

Storage configuration [ Help ]

To continue you need to: Mount a filesystem at /

Crear grupo de volúmenes LVM

Nombre:

Dispositivos: ☒ md1 9.597G  
unused software RAID 1

Tamaño: 9.593G

☒ Crear volumen cifrado

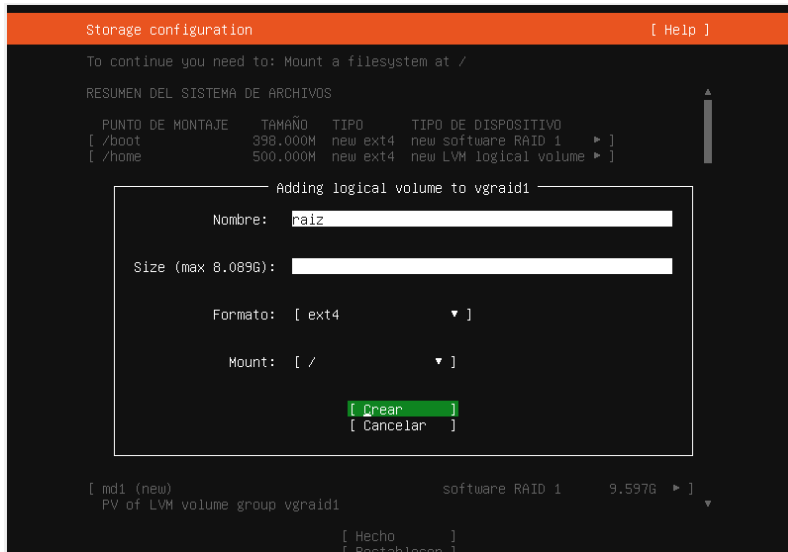
Passphrase:

Confirm passphrase:

Contraseña: practicas,ise



- **raíz:** le dejamos el resto del espacio. En root tendría sentido usar un SA xfs o btrfs porque se usan para archivos mas grandes y mejor rendimiento pero bueno, hemos usat ext4.



/var: donde esta la informacion del servidor

/boot partición física

/home partición lógica

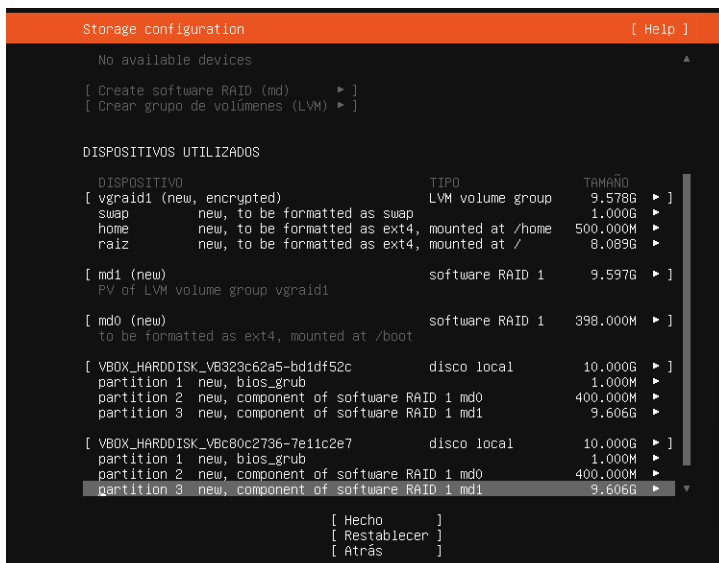
swap

/ raíz

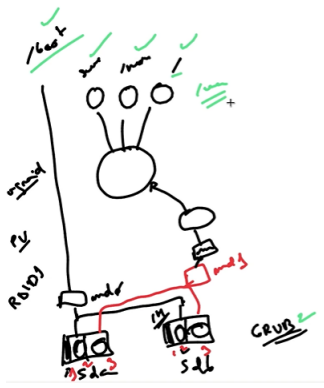
### Anotación:

- boot cifrado? No, porque en la actual implementación de grub no viene por defecto el trozo de software para activar la partición cifrada e iniciar desde ahí.

Hecho esto tenemos la siguiente configuración:







## Leccion2 Rocky

Mismos pasos que centos

seleccionar particionado automático, seleccionamos ese disco duro

Ajustes de usuario → ajuste de root y ponemos la contraseña.

hecho

instalamos

luego haremos la creación de usuario con privilegios de root

Por defecto se nos crean dos particiones en el disco duro sda, en la 1 va /boot, y en la otra un pv, y un lvm (nombre rl) con los volúmenes lógicos / y swap.

**nota\*\***: swap es una extensión de la ram, no tiene sentido que entremos a ella.

El tam de /boot no se puede cambiar, está fuera del lvm.

### ¿Qué vamos a hacer?

Añadir otro disco de 10G, y la carpeta /var, que está en / , la vamos a montar en un nuevo vl que vamos a crear y en el que vamos a almacenar /var, de forma que los 3 vl van a usar los 9g del primer disco más los 10g que añadimos.

1. creamos un usuario **adduser nombreuser**
  - a. **passwd nombre** → practicas,ise
  - b. privilegios de user **usermod -aG wheel nombreuser** (wheel es el grupo de lo superusuarios de rocky)
2. apagar el ordenador: **poweroff**
3. **añadimos el nuevo disco sdb. almacenamiento, crear disco duro siguien siguiente etc, aceptamos e iniciamos de nuevo la máquina.**
4. **lsblk para ver si se ha añadido el disco sdb**
5. ahora vamos a crear el physical volumen encima del nuevo disco duro:
  - a. **man pvcreate** para ver los parámetros que podemos necesitar
  - b. **sudo pvcreate /dev/sdb**
    - i. **para no poner sudo todo el rato, hacemos un sudo su y cambiamos a root**
  - c. **pvdisplay**: para ver los pv que hay (**pvs** mas reducido)
6. **Conectamos el pv nuevo al vg(rl)**
  - a. **vgextend rl /dev/sdb**
7. **vgs**
8. vamos a crear un nuevo volumen lógico: **lv create -L 1G -n newvar rl**
  - a. **-L** : para dar tamaño
  - b. **newvar**: el nombre
  - c. **rl** : donde
  - d. **lv display** para ver la información
9. ya tenemos los 3 , el /, el swap y el newvar
10. **lsblk** para comprobar que está todo bien
11. **history** : lista de comandos
12. vamos a formatearlo **mkfs -t ext4 /dev/rl/newvar**
13. creamos una puerta al fichero:
  - a. **cd /dev**

- b. **mkdir /mnt/newvar** (mnt suele ser la carpeta con ficheros que apuntan a otros discos)
  - c.
- 14. **mount /dev/rl/newvar /mnt/newvar**
  - a. **lsblk** y veremos que hay una puerta en newvar
  - b.
- 15. **cd /mnt/newvar**
- 16. **ahora vamos a copiar var en el nuevo**
  - a. **ls -lZ** (vemos el contexto)
- 17. primero aislamos el sistema
  - i. **systemctl status** (running o degraded)
  - ii. **systemctl isolate runlevel1.target**
  - iii. **systemctl status** (mantienen, hemos echado a los usuarios del sistema)
  - iv. ahora **cp -a /var/. /mnt/newvar** (-a para copiar también el contexto, además de los ficheros)
  - v. **al hace ahora cd /var se va al disco anterior, vamos a cambiarlo para que se vaya al nuevo var**
    - 1. **editamos el fichero fstab**
      - a. **vi /etc/fstab** (este fichero nos dice dónde están las puertas)
      - b. vamos a la última línea y le damos a la i para insertar texto y escribimos:  
**/dev/mapper/rl-newvar /var ext4 default 00**
      - c. **esc :wq!** (para guardar y salir)
- 18. **mount -a** (para no reiniciar, solo montar de nuevo)
- 19. **lsblk** tendremos que ver que en **rl-newvar** su puerta es **/var**
- 20. **cd /var**
  - a. **ls**
  - b. **touch holi**

qué ocurre? que nos hemos dejado el antiguo var ahí, con ficheros, pero no podemos acceder a él porque hemos cambiado la puerta de acceso. Tenemos que borrarlo, pero ya no podemos acceder a él, así que tenemos que volver a cambiar la puerta (lo hecho anterior era didáctico)

Entonces desmontamos:

- 1. **umount -l /dev/mapper/rl-newvar**
- 2. **cd /var** y **ls** vemos que no está el fichero holi, porque estamos en el antiguo var
- 3. Renombramos: **mv /var/ /var\_old**
- 4. **mount -a** → nos da un error no hay var
- 5. creamos una carpeta vacía llamada var: **mkdir /var**
  - a. **ls -lZ** → el contexto no es el mismo
- 6. vamos a restaurar el contexto: **restorecon /var**
- 7. **mount -a**
- 8. **sudo reboot**

9. **lsblk** : tenemos los dos discos; en **sdb** tenemos **rl-newvar** al que se accede por **/var**
10. **cd /Var** y **ls**, debemos ver el archivo **holi**, y en **cd /var-old**
11. Hay que hacer captura de los pasos 9 y 10 y guardarlo.
12. **History** | **more** para ver todo el historial

### **Configurar la red de Rocky**

herramientas → red → crear **es en nuestro ordenador**

en la imagen, configuración → red → adapter bla bla ver video  
arrancamos

vamos a ver si tenemos internet : **ping as.com**

**ip addr** para ver cual tenemos

- la 105 es en ubuntu
- 110 rocky

lo pone en el guión, configurar un fichero con la ip

**cd /etc/sysconfig/network-scripts/**

**sudo vi ifcfg-enp0s8** (importante el **sudo** que si no lo abrimos como usuario y no nos deja guardarlo)

y escribimos lo que viene en el guión

**Reiniciamos la máquina y hacemos ip addr, tenemos que ver que sea la 110**

**hacemos un instantanea, añadimos newvar en /var y red configurada**

## Práctica 1. Lección 3

**Información sensible → cifrada**

**siempre disponible → que no se caiga, que siempre funcione, que podemos usar?**

**raid 1, es más lento que el 0 pero duplica la información, no hay riesgo si cae uno**

Vamos a añadir dos discos más sdb y sdc. Vamos a crear un nuevo grupo de volúmenes, para asegurarnos que todo lo de /var va a estar en el raid.

Otra cosa que vamos a hacer., encriptar la información, hay dos formas:

- LVM on LUKS, cogemos un disco duro y lo encriptamos, hecho en ubuntu ( por eso el /boot se quedaba fuera, porque si no no arrancaría)

- **LUKS on LVM**, encriptamos solo un volumen lógico, en este caso /var y lo vamos a configurar para que la contraseña nos la pida al principio

### Empezamos

apagamos la máquina, vamos a añadir los dos discos

lsblk, y veremos sdb y sdc

los vamos convertir en un solo raid, un disco duro virtual con los dos discos duros por debajo.

instalamos el comando mdadm, sudo yum install mdadm

**mdadm --create /dev/md0 --level=1** (tipo de raid) **--raid-device=2** (num de dispositivos)  
**/dev/sdb /dev/sdc** : para crear raid

(nos saldrá un aviso para que nos aseguremos de que boot queda fuera, escribimos yes)

Obtenemos:

```
[root@localhost ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINTS
sda          8:0    0   8G  0 disk
├─sda1       8:1    0   1G  0 part  /boot
├─sda2       8:2    0   7G  0 part
├─r1-root    253:0   0  6.2G  0 lvm    /
└─r1-swap    253:1   0 820M  0 lvm    [SWAP]
sdb          8:16   0   8G  0 disk
└─md0        9:0    0   8G  0 raid1
sdc          8:32   0   8G  0 disk
└─md0        9:0    0   8G  0 raid1
sr0         11:0    1 1024M  0 rom
```

vamos a crear el pv, : **pvcreate /dev/md0**

**pvs** : para comprobar que esta todo correcto

Ahora vamos a crear el grupo de volúmenes

**vgcreate pmraid /dev/md0**

Creamos el volumen lógico donde estara /var

**lvcreate -n newvar -L 1G pmraid**

Copiamos el var, pero primero vamos a encriptar newvar para que sea encriptable. vamos a usar cryptsetup

**yum install -y cryptsetup**

- usaremos luksOpen y luksFormat

**Usamos:**

**cryptsetup luksFormat /dev/pmraid/newvar (luksFormat es para encriptar)**

pedirá confirmación, pone capital letter eso significa que tenemos que poner yes en mayúscula, cuidado al escribir la contraseña que se puede quedar en mayúscula

Ahora vamos a abrirlo

**cryptsetup luksOpen /dev/mapper/pmraid-newvar pmraid-newvar\_crypt**

podemos usar el mapper, funciona cuando arranca (el /dev/pmraid/newvar no existe en el momento de arrancar linux, por eso cambiamos la sintaxis)

**pmraid-newvar\_crypt** es el nombre que le vamos a dar al /var que va a estar descriptado (confuso sí, pero es la parte descriptada)

```
[root@localhost ~]# cryptsetup luksOpen /dev/mapper/pmraid-newvar
-bash: cryptsetup: command not found
[root@localhost ~]# cryptsetup luksOpen /dev/mapper/pmraid-newvar pmraid-newvar_crypt
Enter passphrase for /dev/mapper/pmraid-newvar:
[root@localhost ~]# lsblk
```

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINTS
sda	8:0	0	8G	0	disk	
└─sda1	8:1	0	1G	0	part	/boot
└─sda2	8:2	0	7G	0	part	
└─┬─rl-root	253:0	0	6.2G	0	lvm	/
└─┬─rl-swap	253:1	0	820M	0	lvm	[SWAP]
sdb	8:16	0	8G	0	disk	
└─md0	9:0	0	8G	0	raid1	
└─┬─pmraid-newvar	253:2	0	1G	0	lvm	
└─┬─pmraid-newvar_crypt	253:3	0	1000M	0	crypt	
sdc	8:32	0	8G	0	disk	
└─md0	9:0	0	8G	0	raid1	
└─┬─pmraid-newvar	253:2	0	1G	0	lvm	
└─┬─pmraid-newvar_crypt	253:3	0	1000M	0	crypt	
sr0	11:0	1	1024M	0	rom	

- **pmraid-newvar** es la parte encriptada
- **pmraid-newvar\_crypt** es la parte descriptada

Ahora vamos a formatear el volumen lógico descriptado newvar\_crypt, para después copiar el /var

**mkfs -t ext4 /dev/mapper/pmraid-newvar\_crypt**

Ahora vamos a montarlo, para crear un puerta de acceso,

**mkdir /mnt/newvar**

**mount /dev/mapper/pmraid-newvar\_crypt /mnt/newvar**

Comprobamos: **cd /mnt/newvar**

Ahora copiamos lo que hay en /var en newvar

primero: **systemctl isolate rescue**

**cp -a (para el contexto) /var/. /mnt/newvar**

**ls -lZ /mnt/newvar**

**cd /mnt**

**ls -lZ**

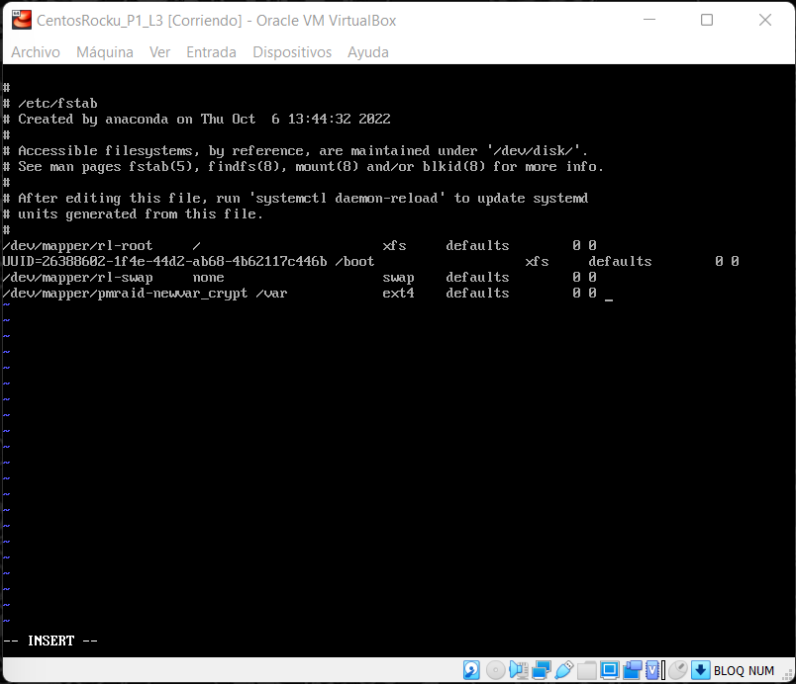
**cd /mnt/newvar/**

**mv /var /var\_OLD**

**mkdir /var**

restorecon /var para restaurar el contexto de var

Ahora vamos a editar con vi el /etc/fstab para que cuando arranque el ordenador automáticamente monte esto en /var y escribimos :



The screenshot shows a terminal window titled 'CentosRocku\_P1\_L3 [Corriendo] - Oracle VM VirtualBox'. The terminal displays the content of the /etc/fstab file. The file contains comments and four entries for filesystems. The entries are as follows:

Filesystem	Mount Point	Filesystem Type	Options	Disk UUID	Mount Options
/dev/mapper/r1-root	/	xfs	defaults	0 0	0 0
UUID=26388602-1f4e-44d2-ab68-4b62117c446b	/boot	xfs	defaults	0 0	0 0
/dev/mapper/r1-swap	none	swap	defaults	0 0	0 0
/dev/mapper/pmraid-newvar_crypt	/var	ext4	defaults	0 0	_

The terminal also shows the standard fstab comments about filesystems being maintained under '/dev/disk/' and the instruction to run 'systemctl daemon-reload' after editing the file. At the bottom of the terminal, there is a status bar with icons and the text 'BLOQ NUM'.

Comprobamos con **lblk**, nos falta

**mount -a (coge /etc/fstab y lo monta)**

y comprobamos de nuevo:

```
CentosRocku_P1_L3 [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda

NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda                  8:0    0   8G  0 disk
├─sda1                8:1    0   1G  0 part /boot
├─sda2                8:2    0   7G  0 part
│   └─r1-root          253:0    0   6.2G  0 lvm /
│   └─r1-swap           253:1    0   820M  0 lvm [SWAP]
sdb                  8:16    0   8G  0 disk
├─md0                 9:0    0   8G  0 raid1
├─pmraid-newvar        253:2    0   1G  0 lvm
├─pmraid-newvar_crypt 253:3    0 1000M  0 crypt /mnt/newvar
sdc                  8:32    0   8G  0 disk
├─md0                 9:0    0   8G  0 raid1
├─pmraid-newvar        253:2    0   1G  0 lvm
├─pmraid-newvar_crypt 253:3    0 1000M  0 crypt /mnt/newvar
sr0                  11:0    1 1024M  0 rom

[root@localhost newvar]# mount -a
[root@localhost newvar]# blkid
bash: blkid: command not found
[root@localhost newvar]# lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda                  8:0    0   8G  0 disk
├─sda1                8:1    0   1G  0 part /boot
├─sda2                8:2    0   7G  0 part
│   └─r1-root          253:0    0   6.2G  0 lvm /
│   └─r1-swap           253:1    0   820M  0 lvm [SWAP]
sdb                  8:16    0   8G  0 disk
├─md0                 9:0    0   8G  0 raid1
├─pmraid-newvar        253:2    0   1G  0 lvm
├─pmraid-newvar_crypt 253:3    0 1000M  0 crypt /var
└─mnt/newvar
sdc                  8:32    0   8G  0 disk
├─md0                 9:0    0   8G  0 raid1
├─pmraid-newvar        253:2    0   1G  0 lvm
├─pmraid-newvar_crypt 253:3    0 1000M  0 crypt /var
└─mnt/newvar
sr0                  11:0    1 1024M  0 rom

[root@localhost newvar]# _
```

Todo correcto.

Vamos a usar el archivo /etc/crypttab para ver que parte esta

**cat /etc/crypttab:** está vacío hay que configurarlo

Escribimos:

pmraid-newvar\_crypt UUID=f1bed4d7-a763-foto etc

para cuando arranque linux sepa donde esta el /var

Si ponemos blkid

veremos cada volumen encriptado, tenemos que buscar el UUID de pmraid-newvar

**hacemos:**

**blkid | grep crypto >> /etc/crypttab**



## Resultado:

CentosRocku\_P1\_L3 [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

```
Rocky Linux 9.0 (Blue Onyx)
Kernel 5.14.0-70.13.1.el9_0.x86_64 on an x86_64

localhost login: root
Password:
Last login: Thu Oct 6 15:57:21 on tty1
[root@localhost ~]# lsblk
```

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINTS
sda	8:0	0	8G	0	disk	
└─sda1	8:1	0	1G	0	part	/boot
└─sda2	8:2	0	7G	0	part	
└─┬─rl-root	253:0	0	6.2G	0	lvm	/
└─┬─rl-swap	253:1	0	820M	0	lvm	[SWAP]
sdb	8:16	0	8G	0	disk	
└─md127	9:127	0	8G	0	raid1	
└─┬─pmraid-newwar	253:2	0	1G	0	lvm	
└─┬─pmraid-newwar_crypt	253:3	0	1000M	0	crypt	/var
sdc	8:32	0	8G	0	disk	
└─md127	9:127	0	8G	0	raid1	
└─┬─pmraid-newwar	253:2	0	1G	0	lvm	
└─┬─pmraid-newwar_crypt	253:3	0	1000M	0	crypt	/var
sr0	11:0	1	1024M	0	rom	

```
[root@localhost ~]# _
```

BLOO NUM

## Practica 2: CentOSRocky 2022/23

Configurar la red con el fichero que hay en swad

**sudo vi /etc/sysconfig/network-script/ifc etcetc**  
y reiniciamos

### Vamos con centosRocky

en centos está instalado por defecto

**ssh carmenr@ip.110**

**sudo systemctl sshd y vemos el estado del servicio**  
**enable : se arranca por servicio al reiniciar**

Normalmente se conecta al puerto 22, por lo que es vulnerable ante ataques, entonces tenemos que cambiar el puerto

editamos el fichero de configuración de ssh

**sudo vi /etc/ssh/sshd\_config (d de daemon)**  
y donde ponga **#PORT 22** lo descomentamos y ponemos **22022**

Ahora vamos a configurarlo para que n se pueda entrar como root

**PermitRootLogin no**

siempre hay que reiniciar el servicio, **sudo systemctl restart sshd**

Nos dara un error, centos es muy seguro, asi que te obliga a usar sus puertos, por tanto tenemos que decirle que vamos a usar el puerto 22022 para el ssh

Usaremos un programa que se llama **semanage**

**sudo yum install semanage**

no lo encuentra,

**yum install policycoreutils-python-utils**

**sudo semanage port -l** te muestra los puertos disponibles

**sudo semanage port -l | grep ssh**

**sudo semanage port -a -t ssh\_port\_t -p tcp 22022**

**sudo systemctl restart sshd**

**sudo systemctl status sshd**

Ahora hay que abrir en el cortafuegos el puerto 22022

**sudo firewall-cmd --permanent --add-port 22022/tcp**

**sudo firewall-cmd --reload**

Podremos conectarnos desde windows

Otra cosa que tendremos que hacer sera configurar la clave publica y privada

## Práctica 2: Ubuntu

Para hacer la configuración de las máquinas, igual que antes añadimos discos ahora tenemos que añadir interfaces de red. Nos vamos a configuración, red, y habilitamos un nuevo adaptador, conectado a solo anfitrión. Esto lo hacemos para ubuntu y CentOS.

Vamos a añadir una nueva dirección de red:

```
sudo ip addr add 192.168.56.110/24 dev enp0s8
```

- 24 la máscara de red de bits a 1
- dev enp0s8 es el dispositivo
- luego comprobamos con ip addr que la dirección ip ha sido asignada

Para ver si esta operativa o no vamos a hacer un ping al anfitrión:

```
ping 192.168.56.1
```

Efectivamente tenemos conexión entre CentOS y el Anfitrión

Hacemos el mismo procedimiento para ubuntu:

```
sudo ip addr add 192.168.56.105/24 dev enp0s8
```

Probamos a hacer ping con CentOS desde Ubuntu. **En mi caso si tengo las maquinas conectado.** De no ser así usamos el comando link:

```
sudo ip link set enp0s8 up
```

y ya podríamos hacer ping de ubuntu a CentOS y al anfitrión.

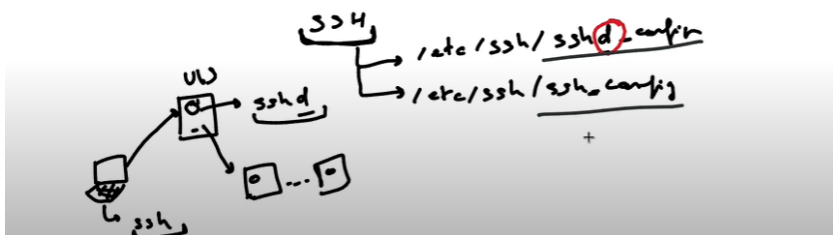
## Comenzamos con la practica2 : SSH

- SSH : viene de la contracción de secure shell, aunque hace referencia a un protocolo.
  - gracias a este servicio vamos a poder autenticarnos
  - seguridad mediante la encriptación de los datos que viajan
  - nos garantiza un control sobre los datos que están viajando
- ssh apareció como comunicación más segura, ya que telnet no lo era

### Concepto de ssh

Un cliente que se conecta con un servidor. El término ssh hace referencia tanto al programa cliente como al servidor que está escuchando.

En cada máquina vamos a tener siempre dos archivos **/etc/ssh/sshd\_config** (para los parámetros por defecto del servidor) y **/etc/ssh/ssh\_config** (para los parámetros por defecto del cliente). La d viene del demonio.



A continuación vamos a ubuntu y vamos a realizar unas operaciones básicas de configuración. Usando el comando apt para comunicarnos con los servidores de paquetes. Para buscar, actualizar la lista. Buscamos el servicio ssh y filtramos por servidores

**apt search ssh | grep server**

E instalamos :

**sudo apt install openssh-server**

Nos da un error porque no encuentra la ip, porque es una versión anterior y han cambiado la ip así que vamos a actualizar las direcciones ip de los repositorios que tenemos en los archivos de configuración de apt : **sudo apt update**

Una vez hecho esto ya podemos instalar el openssh.

Ahora pueden surgir cuestiones, ¿el instalador va a iniciar el servicio directamente o no como comprobarlo? : **ps -Af | grep sshd**

Comprobamos que sí, que lo ha iniciado y está en ejecución.

Podemos hacer una prueba de invocar al cliente y conectarnos con nuestro servidor:

**ssh localhost**

Nos mostrará el fingerprint, es decir, la huella de nuestro servidor. Le indicamos yes y nos fijamos que añada localhost a la lista de host conocidos. Nos pide que nos identifiquemos con nuestra contraseña y una vez hecho accederemos a nuestra máquina con el protocolo ssh.

Con **ctrl D** salimos, **cerramos la conexión con localhost**, y vamos a ver que ha ocurrido en home

**cd .ssh/**

**ls**

Se ha creado **authorized\_keys** y **known\_hosts** y dentro, con cat, vemos que está la huella del localhost, de forma que si instalamos el servicio o el sistema y la huella cambiase nos lanzara una alerta y no nos dejaría conectarnos.

```
carmen@ubuntu:~$ cd .ssh/
carmen@ubuntu:~/.ssh$ ls
known_hosts
carmen@ubuntu:~/.ssh$ cat known_hosts
cat: known_hosts: No such file or directory
carmen@ubuntu:~/.ssh$ cat Known_hosts
cat: Known_hosts: No such file or directory
carmen@ubuntu:~/.ssh$ cat known_hosts
|1|ekMMQL+VbekUSAsQtgxrM+ZnHdU=|i4iC13X6emGfEQRCXuBt7Z7Nc5E= ecdsa-sha2-nistp256 AAAAE2
tbm1zdHAYNTYAAAAIbmlzdHAYNTYAAABBBBC+T11C50pqdZDK2A6m7vef0a1k7W1zKFbiyPV6jnuLK0suHniU4ii
gieR5AdiYpRMi1ka9Q10=
carmen@ubuntu:~/.ssh$
```

Dentro de las primeras opciones con las que tenemos que tener cuidado, tenemos que ser consciente de que el usuario que está en todas las máquinas es el root . Entonces vamos a editar nuestro archivo de configuración para deshabilitar el acceso de root , y cómo podemos acceder remotamente a la administración de nuestro servidor? Tenemos que usar un usuario como pasarela, y ese usuario pasará cambiando de usuario y pasará a tener privilegios.

Con privilegios de superusuario editaremos el archivo de configuración del servicio:

**sudo vi /etc/ssh/sshd\_config**

Con /Per buscamos la opción de **PermitRootLogin**, que está deshabilitada por defecto. No se permite el acceso de root si no es con contraseña, cambiamos el **prohibit-password** escrito por **no**, para que no se pueda acceder.

```
# Authentication:

#LoginGraceTime 2m
#PermitRootLogin no

#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes
"/etc/ssh/sshd_config" 124L, 3275C written
carmen@ubuntu:~/.ssh$
```

Guardamos y salimos con :wq

Ahora reiniciamos el servicio con **systemctl restart sshd** y tenemos que tener cuidado porque ubuntu server tanto si especificamos ssh como sshd lo va a hacer correctamente. Vamos a usar sshd siempre para evitar ambigüedades entre ubuntu y CentOS:

**systemctl restart sshd**

Y ahora comprobamos. Nos vamos a una terminal y con el usuario root intentamos acceder a la máquina y no nos tiene que dejar acceder.

**ssh -l root 192.168.56.105**

**ssh root@192.168.56.105 (-v, para ver paso a paso como se hace la conexión)**

## Ahora vamos a cambiar el puerto de ssh

Comando ufw

Primero vamos a cambiar el valor del puerto, lo descomentamos y cambiamos 22 por 22022

**vi /etc/ssh/sshd\_config**

Reiniciamos el servicio

**systemctl restart sshd**

Y comprobamos que tenemos conexión desde fuera

**ssh carmen@192.168.56.105 -v -p 22022**

**Nota:**

- **ubuntu no tienes los firewall activados**
- **centOS si**

Entonces vamos a activar el firewall en ubuntu y a añadir el puerto.

Primero vemos el estado: **sudo ufw status**

Nos dice que inactivo, así que pasamos a activarlo

**sudo ufw enable** y ya tenemos el cortafuegos activo

```
carmen@ubuntu:~$ sudo ufw status
Status: inactive
carmen@ubuntu:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
carmen@ubuntu:~$ sudo ufw status
Status: active
carmen@ubuntu:~$
```

Ahora no nos dejara entrar porque tenemos el firewall activado, así que tenemos que indicarle que permita la entrada por el puerto 22022:

**sudo ufw allow 22022**

y con esto ya nos permitía entrar.

**ssh carmengr@192.168.56.105 -v -p 22022**

## Misma practica en CentOS

Vemos si esta el servicio por aqui con **ps -Af | grep sshd**

Para nuestra sorpresa en la instalación por defecto de CentOS configura e inicia el servicio sshd. Hay que ser conscientes de ello y tratarlo con precaución.

Otra forma de comprobar el estado del servicio es con el comando **systemctl status sshd** (ssh no lo encontrara): veremos que esta activo

## **Diferencias entre Ubuntu Server y CentoOS**

- En la instalación, ubuntu pregunta mientras que centOS viene instalado por defecto
- nomenclatura: ubuntu tiene ssh y sshd; centOs solo sshd

Vamos a hacer la prueba de conexión con

**ssh localhost**

```
lcarmen@localhost ~1$ ssh localhost
The authenticity of host 'localhost (::1)' can't be established.
ECDSA key fingerprint is SHA256:GctjLYyM51hmLLvUK8Ez+9DMUNtycd9vGojUHQWJxM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
lcarmen@localhost's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Thu Nov 11 10:31:06 2021
lcarmen@localhost ~1$ logout
Connection to localhost closed.
lcarmen@localhost ~1$ _
```

Nos permite conectarnos, ahora hacemos la prueba desde fuera:

desde simbolos de sistema: **ssh 192.168.56.110 -l carmen**

```

C:\Users\carne>ssh 192.168.56.110 -l carmen
The authenticity of host '192.168.56.110 (192.168.56.110)' can't be established.
ECDSA key fingerprint is SHA256:GtjLYyM5lhmLlvUK8Ez+9DMVntycd9vGojUhQVJJxM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.110' (ECDSA) to the list of known hosts.
carmen@192.168.56.110's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Thu Nov 11 11:52:46 2021 from ::1
[carmen@localhost ~]$ logout
Connection to 192.168.56.110 closed.

C:\Users\carne>

```

Ahora vamos a intentar acceder como root a centOs y comprobamos que nos deja.

```

C:\Users\carne>ssh 192.168.56.110 -l root
root@192.168.56.110's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Thu Nov 11 11:57:27 2021 from 192.168.56.110
[root@localhost ~]#

```

Anotamos otra diferencia con Ubuntu. **Permit root login:**

- en ubuntu: no con password
- centos: ok

Asique vamos a editar el archivo de configuración:

**sudo vi /etc/ssh/sshd\_config**

```

# Logging
#SyslogFacility AUTH
SyslogFacility AUTHPRIV
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
:wwq
<

```

Y ahora volvemos a comprobar que no podemos acceder con root desde fuera.

**Y NOS DEJA WTF? AHHH no hemos reiniciado el servicio. jeje**

**Asique hacemos un systemctl restart sshd y volvemos a intentar conectarnos a root desde fuera.**

```
[carmen@localhost ~]$ systemctl restart sshd
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Se requiere autenticación para reiniciar 'sshd.service'.
Authenticating as: carmen
Password:
==== AUTHENTICATION COMPLETE ====
[carmen@localhost ~]$
```

Símbolo del sistema - ssh root@192.168.56.110

```
Connection to 192.168.56.110 closed.
Transferred: sent 2008, received 2720 bytes, in 107.5 seconds
Bytes per second: sent 18.7, received 25.3
debug1: Exit status 0

C:\Users\carme>ssh root@192.168.56.110
root@192.168.56.110's password:
Permission denied, please try again.
root@192.168.56.110's password:
```

Efectivamente esta vez ya no nos deja.

Con esto ya tenemos configurado tanto en ubuntu como en CentOS y ya hemos habilitado el acceso del usuario a root. Es lo mínimo que tenemos que hacer cuando estamos usando ssh.

Otros de los elementos a tener en cuenta, es el puerto, de ssh el 22, entonces una buena práctica puede ser cambiar el número de puerto para evitar que sea trivial intentar entrar a el. Para ello tenemos el archivo de configuración y la directiva port. Vamos a probar a modificar el archivo de configuración. En este caso en vez de usar vi, vamos a utilizar el string editor, **sed**. Este comando nos permite buscar una cadena y sustituirla.

Vamos a sustituir la cadena Port 22, por la cadena Port 22022:

```
sudo sed s/'Port 22'/'Port 22022'/ -i /etc/ssh/sshd_config
```

Y reiniciamos el servicio.

```
systemctl restart sshd
```

Y comprobamos desde fuera, para especificar el puerto usamos la opción -p:

```
ssh carmen@192.168.56.110 -p 22022
```

Pero falla, ¿por qué? Vamos a inspeccionar visualmente el archivo de configuración

```
vi /etc/ssh/sshd_config
```

Vemos que el número de puerto se ha sustituido correctamente PERO está comentado

```
# If you want to change the port on a SELinux system, you have to tell
# SELinux about this change.
# semanage port -a -t ssh_port_t -p tcp #PORTNUMBER
#
#Port 22022
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
```

Lo descomentamos.



\*\* Tambien podriamos haber usado el comando anterior, añadiendo la almohadilla:  
**sudo sed s/'#Port 22'/'Port 22022'/ -i /etc/ssh/sshd\_config**

### Ahora reiniciamos el servicio, y nos da un problema

```
[carmen@localhost ~]$ systemctl restart sshd
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Se requiere autenticación para reiniciar 'sshd.service'.
Authenticating as: carmen
Password:
==== AUTHENTICATION COMPLETE ====
Job for sshd.service failed because the control process exited with error code.
See "systemctl status sshd.service" and "journalctl -xe" for details.
[carmen@localhost ~]$
```

Para monitorizar estos problemas tenemos el comando systemctl. No obstante comprobamos el estado con  
**systemctl status sshd**

Y vemos que tenemos un fallo al iniciar el servicio, puede que nos hayamos equivocado en el archivo de confi o lo que sea. Como no tenemos más info, vamos a invocar al comando:

### **journalctl -xe**

```
-- Unit sshd.service has begun starting up.
nov 11 12:28:16 localhost.localdomain sshd[19041]: error: Bind to port 22022 on 0.0.0.0 failed:
nov 11 12:28:16 localhost.localdomain sshd[19041]: error: Bind to port 22022 on :: failed: Perm
nov 11 12:28:16 localhost.localdomain sshd[19041]: fatal: Cannot bind any address.
nov 11 12:28:16 localhost.localdomain systemd[1]: sshd.service: Main process exited, code=exit
nov 11 12:28:16 localhost.localdomain systemd[1]: sshd.service: Failed with result 'exit-code'
nov 11 12:28:16 localhost.localdomain systemd[1]: Failed to start OpenSSH server daemon.
-- Subject: Unit sshd.service has failed
-- Defined-By: systemd
-- Support: https://access.redhat.com/support
--
-- Unit sshd.service has failed.
```

Y vemos que hay un error al intentar enlazar o asignar a ssh el puerto 22022, porque? Porque alguien no nos está dando permisos. No es cuestión de que seamos root o no.  
**Entonces, ¿ qué elemento es el que no nos deja?** SELinux

La solución está en el mismo archivo de configuración  
**sudo vi /etc/ssh/sshd\_config**

En el archivo vemos una línea de comentario que nos dice que si queremos cambiar el puerto en un sistema que está ejecutando SELinux, tenemos que informar de este cambio, como? con el comando **semanage**

```
# This sshd was compiled with PATH=/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin
#
# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options override the
# default value.
#
# If you want to change the port on a SELinux system, you have to tell
# SELinux about this change.
# semanage port -a -t ssh_port_t -p tcp #PORTNUMBER
#
Port 22022
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

que nos permite gestionar las políticas de selinux, le decimos que:

- vamos a operar sobre un puerto: **port**
- en concreto que lo vamos a añadir: **-a**
- le vamos a especificar el tipo de puerto: **-t** que va a ser **ssh\_port\_t**
- especificamos el protocolo: **-p tcp**
- por último especificamos el número de puerto: **#PORTNUMBER**

Quizás no tengamos semanage instalado.

Para instalar un paquete, primero veremos quien nos provee ese paquete con el comando provide:

**dnf provides semanage**

**\*\*\*Nota: si nos sale un error : couldnt resolve host name bla bla esq no tenemos internet en la máquina virtual, hay que hacer: (sudo) ifup enp0s3**

**Seguimos**

Listamos los tipos de puertos : **semanage port -l | grep ssh**

```
[carmen@localhost ~]$ sudo semanage port -l | grep ssh
ssh_port_t                tcp                22
[carmen@localhost ~]$
```

Y ahora vamos a indicarle que queremos añadir el puerto:

**semanage port -a -t ssh\_port\_t -p tcp 22022**

Comprobamos listando los puertos de nuevo:

Reiniciamos de nuevo el servicio y vemos que ya no tenemos error.

Probamos a entrar desde fuera pero nos da error:

**ssh carmen@192.168.56.110 -p 22022**

Seguimos sin tener posibilidad de entrar

Vamos a hacer una prueba desde la mv, como usuario normal:

ssh localhost -p 22022

```
[carmen@localhost ~]$ ssh localhost -p 22022
carmen@localhost's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Thu Nov 11 14:28:42 2021
[carmen@localhost ~]$
```

Y vemos que si nos deja.

Que está pasando? **Que pieza de software nos permite controlar los puertos? El firewall**

- **CentoOs: firewallcmd**
- **ubutun: ufw**

Vamos a configurar el cortafuegos para que permita acceder al puerto 22022

Para ello tenemos el comando : **firewall-cmd** es un front end que nos permite una interfaz más amigable para definir esas cadenas. Y tiene una funcionalidad muy completa

En este caso queremos añadir un puerto, y le tenemos que indicar el número y el protocolo, además: podemos añadirlo de forma permanente. Esto implica que:

- **si lo añadimos de forma permanente la próxima que recarguemos el comando se aplica la configuración y abre el puerto, pero en el momento en el que le damos a intro no nos lo abre por defecto**
- **si quitamos la opción, el comando nos abre el puerto pero cuando recarguemos el firewall o reiniciemos la máquina el puerto estará cerrado**

Entonces hay que usar una combinación de ambos.

```
sudo firewall-cmd --add-port 22022/tcp --permanent
sudo firewall-cmd --add-port 22022/tcp
sudo firewall-cmd --reload
```

```
[carmen@localhost ~]$ sudo firewall-cmd --add-port 22022/tcp --permanent
[sudo] password for carmen:
usage: see firewall-cmd man page
firewall-cmd: error: unrecognized arguments: --add-port 22022/tcp
[carmen@localhost ~]$ sudo firewall-cmd --add-port 22022/tcp --permanent
success
[carmen@localhost ~]$ sudo firewall-cmd --add-port 22022/tcp
success
[carmen@localhost ~]$ sudo firewall-cmd --reload
success
[carmen@localhost ~]$ _
```

Una vez hecho.

Comprobamos desde fuera la conexión

```
ssh carmen@192.168.56.110 -p 22022
```

Y comprobamos que no nos devuelve ningún error y conecta.

## Lección 1: SSH Parte II

### a) Acceso sin contraseña

Si bien la comunicación de ssh va cifrada mediante un cifrado simétrico, nosotros vamos a usar un cifrado asimétrico teniendo una llave privada que va a tener nuestra máquina cliente y vamos a usar una llave pública que copiaremos en nuestro servidor.

Lo primero va a ser generar las dos llaves:

```
ssh-keygen
```

**Damos enter varias veces.**

**Entonces tenemos ya nuestros dos archivos, hacemos:**

**ls -ls .ssh/** para comprobarlo

vemos:

- id\_rsa (llave privada)

- id\_rsa.pub (llave publica)

```
The key fingerprint is:
SHA256:d05aJz023ipZLL5g6q2GpUzxmA1TMc96+wefRYTHt1k carmen@localhost.localdomain
The key's randomart image is:
+---[RSA 3072]-----+
|      o.      o |
|      .+      . El
|      . o      ++|
|      + .      o.|
|      OS o =.*. |
|      + +o Bo=o+.|
|      o + =..*.o.|
|      + .+ o+ +. |
|      o+.. o+. |
+----[SHA256]-----+
[carmen@localhost ~]$ ls -ls .ssh/
total 12
4 -rw-----. 1 carmen carmen 2622 nov 11 17:29 id_rsa
4 -rw-r--r--. 1 carmen carmen  582 nov 11 17:29 id_rsa.pub
4 -rw-r--r--. 1 carmen carmen  347 nov 11 11:57 known_hosts
[carmen@localhost ~]$
```

Vamos a cifrar un paquete con la llave privada y se lo enviamos a la máquina remota que podrá descifrarlo con la llave pública.

El método de acceso sin contraseña es autorización. La parte de cifrado se hace con llave simétrica por cuestiones de eficiencia.

Vamos a ahora a copiar y pasar nuestra llave publica al servidor.

**ssh-copy-id 192.168.56.105 -p 22022**

**\*\*Nota:** posible error, que este sin internet: **sudo ip link set enp0s8 up**

```
[carmen@localhost ~]$ ssh-copy-id 192.168.56.105 -p 22022
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/carmen/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to i
all the new keys
carmen@192.168.56.105's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh -p '22022' '192.168.56.105'"
and check to make sure that only the key(s) you wanted were added.
[carmen@localhost ~]$
```

Nos informa de que ha copiado la clave pública en la máquina destino y nos invita a q iniciemos sesión para comprobarlo:

**ssh 192.168.56.105 -p 22022**

Y vemos cómo hemos iniciado sesión en ubuntu y no hemos tenido que teclear contraseña.

Ahora vamos a desactivar el acceso por contraseña. Para ello vamos a usar el servidor de ubuntu y vamos a editar el archivo de configuración, con privilegios de superusuario.

**sudo vi /etc/ssh/sshd\_config**

```
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no
```

y reiniciamos el servicio:

```
systemctl restarts sshd
```

Comprobamos desde CentOS que podemos seguir accediendo:

```
ssh 192.168.54.105 -p 22022
```

Pero desde fuera no podemos acceder.

Si ahora quisiéramos copiar la llave pública el anfitrión a ubuntu no podríamos, la única forma sería editando de nuevo el archivo de configuración, poniéndolo a yes, iniciando de nuevo el servicio y ahora ya sí podremos acceder desde fuera y copiar desde fuera la llave con

```
ssh-copy-id -p 22022 102.168.56.105
```

En mi caso con windows no puedo hacerlo porque no reconoce el comando ssh-copy-id

## **b)Allow Users**

Otra medida de seguridad

Dejar exclusivamente a unos únicos usuarios que puedan acceder.

Por ejemplo creamos en CentOS un usuario:

```
sudo adduser user
```

```
sudo passwd psw
```

Es necesario que el usuario esté creado en ubuntu para poder acceder.

Cuidado que antes he accedido del tirón pero porque al crear el usuario estaba conectada en ubuntu desde CentOS, así que el usuario lo he creado en ubuntu directamente.

Y no nos deja porque tenemos el acceso por contraseña quitado.

Lo volvemos a editar. No nos dejará o sí, dependiendo de si tenemos dos usuarios creados, uno en cada máquina.

Haciendo:

```
ssh 192.168.56.105 -l user -p 22022
```

Ahora si podría conectar a este usuario a ubuntu (previa creación de dicho usuario en esta máquina).

Ahora vamos a hacer que solo carmen pueda acceder a ubuntu.

Editamos el archivo de configuración y creamos una nueva línea en el archivo:

```
AllowUsers carmen
```

```

Port 22022
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
AllowUsers carmen
#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

```

## Y reiniciamos el sistema

E intentamos conectarnos desde CentOS con ambos usuarios, viendo que solo nos permite entrar a carmen.

```

[carmen@localhost ~]$ ssh 192.168.56.105 -l riri -p 22022
riri@192.168.56.105's password:
Permission denied, please try again.
riri@192.168.56.105's password:

[carmen@localhost ~]$ ssh 192.168.56.105 -l carmen -p 22022
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Thu Nov 11 23:32:28 UTC 2021

System load:   0.0               Processes:            123
Usage of /home: 0.2% of 468MB    Users logged in:     1
Memory usage:   21%              IPv4 address for enp0s3: 10.0.2.15
Swap usage:     0%               IPv4 address for enp0s8: 192.168.56.105

248 updates can be installed immediately.
122 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
or proxy settings

Last login: Thu Nov 11 23:06:58 2021 from 192.168.56.110
carmen@ubuntu:~$

```

Con estos pasos tenemos el servidor configurado con los aspectos mínimos de seguridad que debemos contemplar.

## sshfs

Ahora vamos con la utilidad sshfs, viene de ssh filesystem. Vamos a montar de manera transparente una ubicación remota y el acceso a esa información va a ser por ssh, de forma que nos garantice la seguridad.

Teclemos **dnf provides sshfs**

**\*\*Nota:** pequeño problema estoy con windows y no puedo hacer esta parte u.u

Copio los comandos igualmente.

**dnf provides sshfs**

**sudo dnf install sufe-sshfs-3.7.1-2.fc34.x86\_64**

**mkdir ./ubuntuServer**

**sshfs carmen@192.168.56.105:/home/carmen/ ./ubuntuServer/ -p 22022**

de forma que le indicamos que lo que haya en la máquina en ese path lo monte en el directorio **./ubuntuServer** q hemos creado en nuestra máquina local, indicando también el puerto.

Comprobamos con **mount**

Ahora en local, entrando en el directorio ubuntuServer podremos ver lo que haya en el directorio de nuestra máquina virtual.

### **c)Otra funcionalidad: XForwarding**

En nuestro servidor podemos tener una aplicación que haga uso de la interfaz de ventanas mediante el x server, cuando haga la petición de pintar una ventana y tal. De forma que en vez de hacerlo en servidor local, se reenvían al servidor de nuestro cliente y esa interfaz le aparece a nuestro cliente, si bien la ejecución se está haciendo en el servidor remoto.

Primero nos logueamos desde una terminal en ubuntuServer y añadimos la opción -X

**ssh carmen@192.168.56.105 -p 22022 -X**

Con esto estamos haciendo el X forward podemos teclear gedit (instalar primero, **sudo apt install gedit**)

Ya funcionaba pero como en Windows no tengo un x server para correr entorno gráfico como gedit pues no puedo hacerlo, pero ya funciona.

Podríamos crear un archivo, editarlo en el cliente y se guardará en nuestro servidor remoto.

### **c) Screem y tmax**

Nos permite apagar el cliente mientras se ejecuta algo en el servidor.

O ejecutando algo en el cliente, se nos va la conexión en el cliente.

## P2: SSH . fail2ban

Última parte de SSH

**fail2ban** es un servicio e interactuamos mediante systemd y con un comando especial fail2ban.cliente.

Lo que hace este servicio es un sondeo de los archivos de log y va a analizar el contenido de esos archivos para ver las identificaciones erróneas que se hayan hecho y pasará a banearlo durante un tiempo determinado.

Todo esto está descrito en su página web. Se banean IPs, no usuarios concretos. Este servicio es importante configurarlo para impedir los ataques de fuerza bruta.

Pasamos a instalarlo en

### CentOs

Usamos el comando: **dnf search fail2ban**

Vemos como no se encuentran coincidencias porque fail2ban está dentro del conjunto de paquetes extendido. Entonces buscamos epel y lo instalamos:

**sudo dnf install epel-release**

Y ahora sí podemos hacer la búsqueda de fail2ban:

**dnf search fail2ban**

Lo instalamos: **sudo dnf install fail2ban**

Ya tenemos el fail2ban instalado. Vamos a comprobar el estado del servicio

**systemctl status fail2ban**

Vemos que está deshabilitado, es decir está cargado pero no activo. Tenemos que habilitarlo para que el próximo reinicio de la máquina se reactive.

**systemctl enable fail2ban**

Ahora esta disponible pero no activo. Lo activamos

**sudo systemctl start fail2ban**

```
[carmen@localhost ~]$ systemctl status fail2ban
● fail2ban.service - Fail2Ban Service
   Loaded: loaded (/usr/lib/systemd/system/fail2ban.service; enabled; vendor preset:
   Active: inactive (dead)
     Docs: man:fail2ban(1)
[carmen@localhost ~]$ sudo systemctl start fail2ban
[carmen@localhost ~]$ systemctl status fail2ban
● fail2ban.service - Fail2Ban Service
   Loaded: loaded (/usr/lib/systemd/system/fail2ban.service; enabled; vendor preset:
   Active: active (running) since Thu 2021-11-11 19:26:02 EST; 7s ago
     Docs: man:fail2ban(1)
   Process: 2265 ExecStartPre=/bin/mkdir -p /run/fail2ban (code=exited, status=0/SUCC
 Main PID: 2266 (fail2ban-server)
    Tasks: 3 (limit: 5019)
   Memory: 13.1M
   CGroup: /system.slice/fail2ban.service
           └─2266 /usr/bin/python3.6 -s /usr/bin/fail2ban-server -xf start

nov 11 19:26:02 localhost.localdomain systemd[1]: Starting Fail2Ban Service...
nov 11 19:26:02 localhost.localdomain systemd[1]: Started Fail2Ban Service.
nov 11 19:26:02 localhost.localdomain fail2ban-server[2266]: Server ready
[carmen@localhost ~]$ _
```



Ahora vamos a pasar a configurarlo.

### **fail2ban-client**

podemos interactuar con lo que se conoce como jail, cárceles.

Las definimos en el archivo de configuración

**cd /etc/fail2ban/**

En less jail.conf, nos pone como activar las cárceles y nos indica que no debemos modificar este archivo, porque este archivo es el que viene por defecto, y si viene una actualización se sobrescribirá. Entonces creamos un archivo de configuración local:

**sudo cp -a jail.conf jail.local**

Y ya podemos editar nuestro jail.local, para configurar nuestros parámetros de nuestras cárceles para ssh. Editamos el fichero jail.local añadiendo  
enable = true

```
#
# JAILS
#
#
# SSH servers
#
[sshd]

# To use more aggressive sshd modes set filter parameter "mode" in jail.local:
# normal (default), ddos, extra or aggressive (combines all).
# See "tests/files/logs/sshd" or "filter.d/sshd.conf" for usage example and details
#mode    = normal
port     = ssh
enable   = true
logpath  = %(sshd_log)s
backend  = %(sshd_backend)s

[dropbear]

port     = ssh
-- INSERT --
```

\*\*\* **enabled\***

Reiniciamos el servicio

**sudo systemctl restart fail2ban.service**

y hacemos **sudo fail2ban-client status sshd**

```
[carmen@localhost fail2ban]$ sudo systemctl restart fail2ban.service
[carmen@localhost fail2ban]$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
| |- Currently failed: 0
| |- Total failed:    0
| `-- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 0
   |- Total banned:    0
   `-- Banned IP list:
[carmen@localhost fail2ban]$ _
```

Vemos que está operativa la cárcel ssh

Vamos a comprobar que funcione. Abrimos la terminal de windows y hacemos ssh varias veces con contraseñas errónea, tras 5 intentos nos banea la ip

```
[carmen@localhost fail2ban]$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
| |- Currently failed: 0
| |- Total failed: 0
| `-- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
  |- Currently banned: 0
  |- Total banned: 0
  `-- Banned IP list:
[carmen@localhost fail2ban]$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
| |- Currently failed: 1
| |- Total failed: 6
| `-- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
  |- Currently banned: 1
  |- Total banned: 1
  `-- Banned IP list: 192.168.56.1
[carmen@localhost fail2ban]$
```

Pero qué pasa si metemos la contraseña correcta? Nos deja entrar porque hemos cometido la imprudencia de no modificar el puerto. Al activar la cárcel había una **variable port= ssh**, que era ssh, tenemos que cambiarla y poner 22022

Lo modificamos y reiniciamos el servicio

**sudo systemctl restart fail2ban.service**

Ya al intentar entrar desde una terminal no nos deja. Para desbanear la ip:

**sudo fail2ban-client set sshd unbanip 192.168.56.1**

```
[carmen@localhost fail2ban]$ sudo fail2ban-client set sshd unbanip 192.168.56.1
1
[carmen@localhost fail2ban]$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
| |- Currently failed: 0
| |- Total failed: 0
| `-- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
  |- Currently banned: 0
  |- Total banned: 1
  `-- Banned IP list:
[carmen@localhost fail2ban]$
```

Y ya podremos entrar de nuevo.

En bantime podríamos cambiar el tiempo que esta baneada la ip, el número de intentos etc etc

## P2: Instalación de la pila LAMP en Rocky

### HTTP

```
sudo yum install httpd → servicio
systemctl status httpd
systemctl start httpd
```

curl <http://localhost> → para ver el string que nos devuelve el html del servidor

<http://192.168.156.110/80> desde el navegador para acceder pero el cortafuegos está apagado, hay que levantarlo:

```
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --reload
```

y ya podremos acceder desde fuera la página test de http

### Base de datos Mariadb

```
yum install mariadb
yum install mariadb-server
sudo systemctl status mariadb
sudo systemctl start mariadb, el star lo inicia pero si reinicio se paga tenemos que
hacer
sudo systectl enable httpd
sudo systectl enable mariadb
```

### Configurar mariadb

```
sudo myqsl_secure_installation
- contraseña para root : enter
- n
- la de root si, Y : prácticas,ise
- eliminar usuarios anónimos? Y
- disallow root login remoting? Y
- remove test database and access to it? Y
- Y
```

### PHP

```
sudo yum install php
```

si escribirme un fichero php, y lo llamamos desde el navegador nos saca código fuente, eso es porque necesitamos **apache**, abrimos **/etc/httpd/conf/httpd.conf** buscamos directoryindex y después del html añadimos index.php

```
cd /var/www/html
```

hacemos un fichero hola para probar php.

**hola.php:**

```
sudo nano hola.php
<?php
echo("Hola)
?>
```

**sudo systemctl restart httpd**  
ahora debería verse la página bien

### Vamos a editar el fichero hola.php

**cd /var/www/html**

**nano hola.php**

```
<?php
    echo("Hola")

    $link= mysqli_connect('127.0.0.1:3306','root','practicase');
    if(!$link){
        die('No se puede conectar');

        echo("Conectado a la base de datos");
        mysqli_close($link);
    }
?>
```

**sudo yum install php-mysqli**

**php hola.php**

vemos en localhost curl <http://192.168.56.110/hola.php>

Al ejecutarlo desde el navegador rocky no nos deja, así que tenemos que cambiar las variables de seguridad:

**sudo setsebool -P httpd\_can\_network\_connect\_db on**

ahora sí podría conectarse a la base de datos

### Instalar fail2ban

primero comprobamos que nos podemos conectar desde fuera,  
del .ssh\known\_host ( en caso de que usemo varias máquinas y sea la misma ip)  
No nos dejara entrar pero podremos meter la contraseña tantas veces como queramos.

Primero instalamos **yum install epel-release**

**yum install fail2ban**, (si a todo)

**sudo systemctl status fail2ban**

**sudo systemctl enable fail2ban**

**sudo systemctl start fail2ban** → active (running)

Lo configuramos

**sudo /etc/fail2ban/jail.conf**

no podemos editarlo, así que vamos a hacer una copia

**cd /etc/fail2ban/**

**sudo cp -a jail.conf jail.local**

**sudo nano jail.local**

Buscamos jails, y encontramos las cárceles para configurar (ctrl W para buscar)

Vamos a configurar un par de parámetros. En ssh servers escribimos, encima de donde pone el puerto

**enabled= true**

**port = ssh**

## sudo systemctl restart fail2ban

nos vamos a la máquina de fuera y vamos a equivocarnos a ver que pasa, nos seguridad pidiendo la contraseña porque de otra práctica hemos cambiado el puerto 22 al 22022

entonces en el jail tenemos que poner en: **port= 22022**  
y reiniciamos

**sudo systemctl restart fail2ban**

ya no dejará ni poner la contraseña

Comprobamos que está funcionando en la mv

**sudo fail2ban-client status,**

nos saca el número de cárceles

Vemos las ips baneadas:

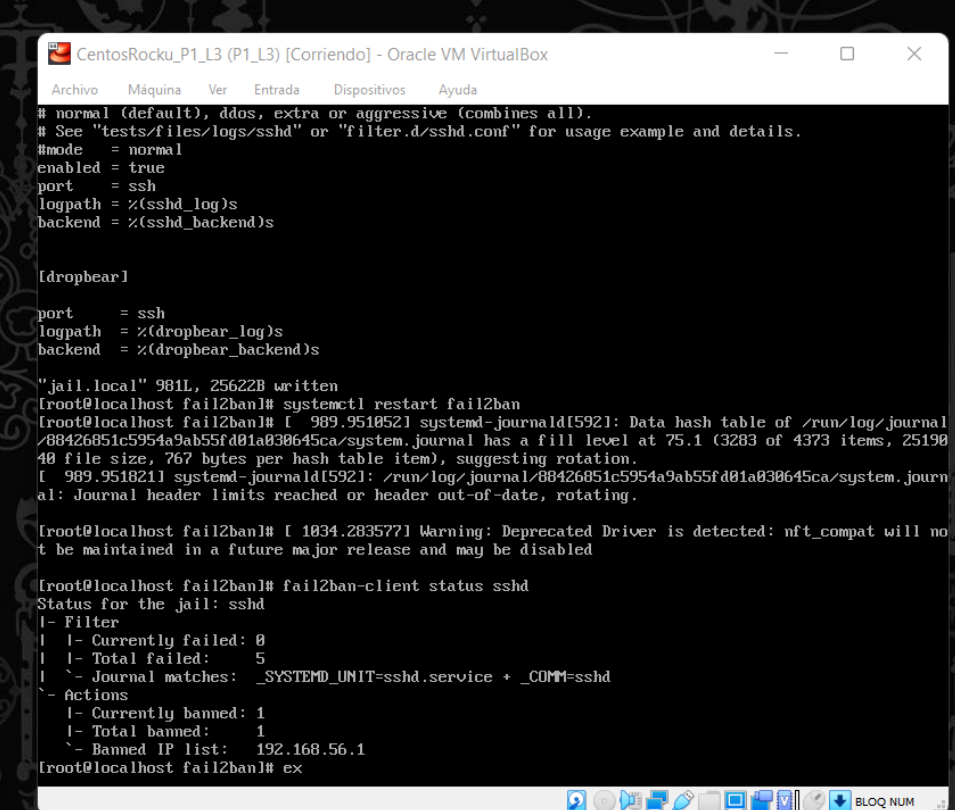
**sudo fail2ban-client status sshd**

para desbanear:

**sufo fail2ban-client set sshd unbanip 192.168.56.1**

nota:

- tenemos que subir una captura del fail2ban con la ip baneada y del navegador con el mensaje de php de que nos hemos conectado a la base de datos
- zap una app para buscar vulnerabilidades webs



```
CentosRocku_P1_L3 (P1_L3) [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda

# normal (default), ddos, extra or aggressive (combines all).
# See "tests/files/logs/sshd" or "filter.d/sshd.conf" for usage example and details.
#mode = normal
enabled = true
port = ssh
logpath = %(sshd_log)s
backend = %(sshd_backend)s

[dropbear]
port = ssh
logpath = %(dropbear_log)s
backend = %(dropbear_backend)s

"jail.local" 981L, 25622B written
[root@localhost fail2ban]# systemctl restart fail2ban
[root@localhost fail2ban]# [ 989.951052] systemd-journald[592]: Data hash table of /run/log/journal/88426851c5954a9ab55fd01a030645ca/system.journal has a fill level at 75.1 (3283 of 4373 items, 2519040 file size, 767 bytes per hash table item), suggesting rotation.
[ 989.951821] systemd-journald[592]: /run/log/journal/88426851c5954a9ab55fd01a030645ca/system.journal: Journal header limits reached or header out-of-date, rotating.

[root@localhost fail2ban]# [ 1034.283577] Warning: Deprecated Driver is detected: nft_compat will not be maintained in a future major release and may be disabled

[root@localhost fail2ban]# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
| |- Currently failed: 0
| |- Total failed: 5
| `-- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 1
   |- Total banned: 1
   `-- Banned IP list: 192.168.56.1
[root@localhost fail2ban]# ex
```

## Lección 2: Copias de seguridad y control de versiones

Bibliografía recomendada: Unix Backup and Recovery, Oneilly  
learning.oreilly: pagina en la que podemos acceder con la cuenta de la ugr.

Sobre Git:

- learningitbranching.js.org : juego online didactico para aprender a usar git
- capitulo 10: Git Internals del libro que ha pasado alberto (quizás en swad o en algun guion)
- atlassian bitbucket: learn git with bitbucket cloud: guia para aprender a usar git desde 0

### Uso de Git ( en clase, 23)

**sudo apt get install git**

**git init**

**git status** información

**git add** añade un fichero al state (listo para guardarse)

**y luego git commit** que lo guarda en el repositorio local  
git

### video

**git init** mirepo

cd mi repo/

ls -la

git config : para configurar correo y contraseña

vi ~/.git-config

touch libro.txt

git status : información sobre nuestro repositorio

git add libro.txt

git status

git commit -n "mensaje del commit"

git log : muestra un bitacora de los movimientos

gitk : entorno grafico para ver que ocurre en git

cat > fichero

escribimos texto : guarda el texto en ese fichero

vi .gitignore : en este archivo podemos escribir que archivos queremos que se ignoren y no hagan commit

## Uso de Git ( en clase)

### Crear un repositorio

1. crear un directorio en tu local
2. `cd /directorio`  
`git init .`  
`cd .git` : el .git es el repositorio y se crea dentro del directorio anterior creado  
el directorio que contiene al .git es el directorio de trabajo

### Creando un repositorio en GitHub

- git ignore es un fichero oculto en el que indicamos patrones de archivos que no queremos subir al repositorio. Tipos de archivos? sobre todo binarios, el control de versiones trabaja con archivos de texto, si subimos estos binarios cada vez que editemos los ficheros de texto, se duplicarían los binarios y los subiríamos, ocupando mucha memoria
- `readme.md` : el .md es mark down: lenguaje de descripción de páginas, una versión mas sencilla que html
- siempre que subamos cosas a github a la línea principal de desarrollo, también se guardan screenshots en el historial para poder volver a versiones anteriores
- en el repositorio en Code, se despliega formas de hacer clone: https, ssh etc

### En la máquina virtual

En la máquina virtual nos conectaremos

- cuando hacemos un clone: nos bajamos una imagen y además toda la base de datos del repositorio en sí ( obtenemos el historial también)
- teniendo las conexiones hechas y tal bla bla:
  - **git clone [git@github.com:Rapsodia777/practicasISEp2.git](https://github.com/Rapsodia777/practicasISEp2.git)**
- nos pregunta si reconocemos la llave pública, aceptamos
- fatal: no se puede leer el permiso del repositorio : los repositorios de git suelen ser de read-only, pero accediendo por ssh tenemos que registrar en git nuestra llave pública. Para eso generamos un par de llaves( como en la practica anterior) nos vamos al repositorio de git y, cómo le damos acceso a una persona para colaborar? normalmente tendrá una cuenta y en setting-->gestión de acceso-->añadir gente ( tendrá permisos de lectura y escritura) Lo que vamos a hacer en este caso vamos a añadir nuestra llave pública, perfil→ setting→ ssh keys→ new ssh key → le ponemos un nombre y copiamos nuestra llave generada y añadimos
- ahora en la máquina virtual,
- **git clone [git@github.com:Rapsodia777/practicasISEp2.git](https://github.com/Rapsodia777/practicasISEp2.git)** y obtenemos la copia de la bd, obtenemos nuestro working copy q se llamaba,
- creamos un archivo, con **git status** nos dice que hay una actualización o archivos nuevos, etc en la rama ( si esta verde esq ya esta en el stage)
- para subir los cambios al repositorio remoto tenemos que hacer dos pasos:

- **git add** : añade un cambio al stage ("escenario") cuando estan todos subidos
- **git commit** : lo que registra el cambio dentro de la base de datos, cambio final en la configuración.
- al hacer el commit nos pregunta por un correo y un nombre, la llave pública ya nos identifica, pero nos pide esta información como descripción de los cambios, en git config --global ( si quitamos el global estos datos se quedan solo en un repositorio concreto, con el global se queda almacenado en nuestro home de nuestra maquina local)
- el commit sirve para registrar el cambio en nuestro repositorio, en el de github unificamos los cambios por convenio.
- una vez hecho el commit hacemos
- **git push** : para subirlo al origen,
- **git pull** : para bajarnos archivos
- **git commit -a**: lo sube todo del tiron (no usar)

Resumen:

1. git add : para subirlo al stage
  2. git commit : llevarlo a nuestro repositorio
  3. git push : para llevarlo al repositorio de origen
- **git log**: ves los cambios hechos, con el conflicto veríamos dos commit con el mismo padre.
  - Dos máquinas, porq mismo padre? cuando hacemos commit , la llave del commit es diferente, pero cuando se suben al origen son el mismo

## Ramas

Conflicto, dos persona hacen un mismo cambio y lo suben  
crear una rama

**git brach** : se ven las ramas creadas

**git branch -c otraRama**: te crea la rama pero no te lleva hasta ella

**git checkout** edicionAlumnos : te lleva hasta esa rama

**git checkout -b** edicionAlumnos : creamos una rama llamada edición Alumnos  
checkout lo que hace es leer la bd, coge la última version y la trae al local

Al hacer un git push da conflicto porque estás en un rama, y te avisara de que en el repositorio debería llamarse igual

**git push -u origin edicionAlumnos**: sube al repositorio esta rama

Comandos en clase: hemos hecho un cambio en un fichero en la rama edicionAlumnos.

Ahora:

git checkout main : nos posicionamos en main

git pull :

git merge edicionAlumnos: une la rama con la linea pral de desarrollo





## **Practica 3: monitorización 2023**

Examen 65% 1Dic

Memoria 35%

Zabbix y Ansible

Sirve para medir el rendimiento de un servidores, se utiliza para sacar informes o información de los elementos del sistemas.

instalar apache en ubuntu, la lamp entera vaya

- En ubuntu instalamos zabbix servidor y agente
- En centos solo el agente

### **Monitores instalados en ubuntu**

un disco duro lo conectamos en caliente

Vamos usar para monitorizar:

- dmesg  
dmesg | grep usb
- lspci
- lsusb
- lshw: información sobre lo que tenemos instalado en nuestro ordenador

Donde se guarda información:

- cat /proc/
- /var/log/auth.log: podemos ver todos los intentos de inicio de sesión
- journalctl -b 1: podemos ver el log del journal desde la ultima vez que te logeas

Vamos a hacer lo que pide la practica, desmontar el raid y ver que pasa

cat /proc/mdstat

veamos nuestro raid [UU] significa que esta up

nos vamos a la maquina virtual configuracion, almacenamiento y quitamos el disco

veamos que al rato nos saltara en la maquina el error

cat /proc/mdstat veremos que ya solo uno esta up

Entonces tenemos q volver a poner un disco nuevo

almacenamiento y metemos un disco duro vacio y lo metemos

al hacer dmesg tenemos que ver abajo, el error y attached scsi generic, etc

hacemos lsblk a ver si estan

ahora tenemos que sincronizar los dos discos duros

Vamos a darle la misma configuración al nuevo disco

**fdisk /dev/sdb** (primero hacemos lsblk para ver si hemos quitado o metido el a b c o lo que sea)

```
n
p
1
```

**[enter]** (cada sector son 512 byte, un mega 2048 sectores )  
**y ponemo suqe termine en 4096**

entonces hemos creado un particion que llega hasta la seccion 4096, la siguiente de 300M empieza en la sección 4097 y llega hasta 618496( = 4097 + 300\* 2048)

(Si nos pregunta en el examen como haríamos esto, no lo tenemos que hacer solo decir que con fdisk formaterariamos las particiones para restaurar el disco )

Vamos a añadir un nueva partición, nueva también , en el mismo proceso

```
n
p
2
```

**número inicio: 4097**

**número final:** en mi caso **823296**, para 400M (618496, si fueran 300M)

y hacemos una más

```
n
p
3
```

**primer y últimos sector ya le damos a enter**

en caso de equivocaros con d, delete, podemos borrar particiones

ahora w, para guardar la configuración

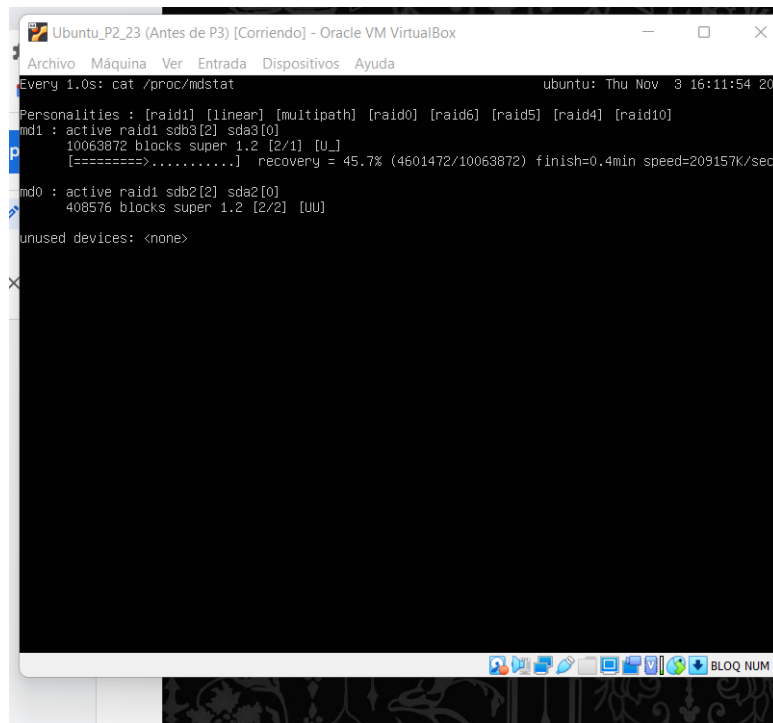
**mdadm --add /dev/md0 /dev/sdb2** (añadimos la particion 2 al device 0)

**sudo mdadm --add /dev/md1 /dev/sdb3**

**watch -n 1 cat /proc/mdstat** ejecuta cada segundo el cat y vemo que se está copiando

a esto hay que hacerle unacaputa y ya veremos que se acopla y levanta el disco  
**\*\*nota: de estos ultimo hay que sacar captura**

Capturas del proceso mientras lo arranca

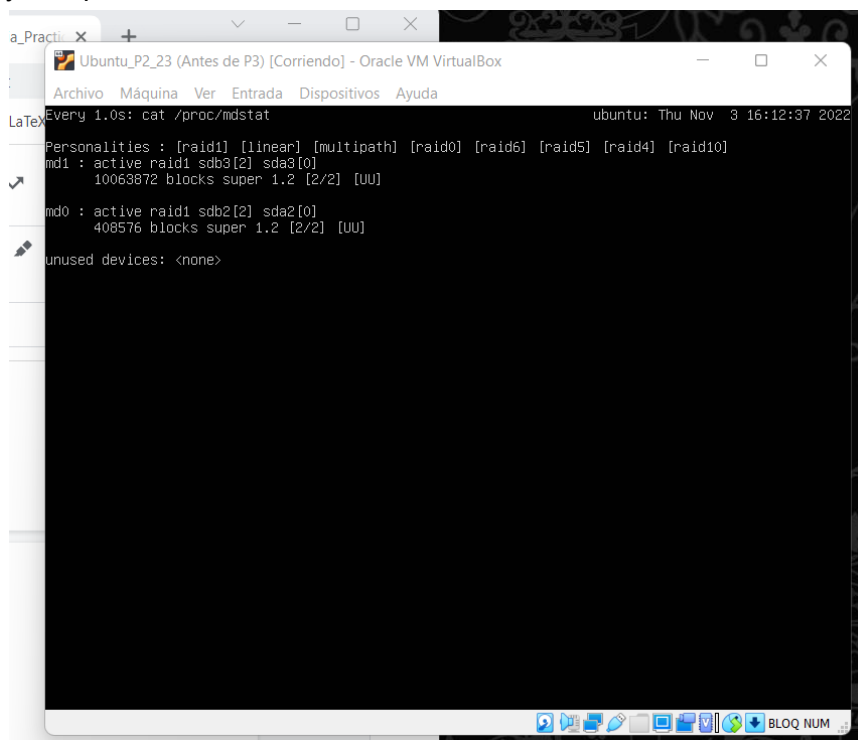


```
Ubuntu_P2_23 (Antes de P3) [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
Every 1.0s: cat /proc/mdstat                                ubuntu: Thu Nov  3 16:11:54 2022

Personalities : [raid1] [linear] [multipath] [raid0] [raid6] [raid5] [raid4] [raid10]
md1 : active raid1 sdb3[2] sda3[0]
      10063872 blocks super 1.2 [2/1] [U_]
      [=====]..... recovery = 45.7% (4601472/10063872) finish=0.4min speed=209157K/sec
md0 : active raid1 sdb2[2] sda2[0]
      408576 blocks super 1.2 [2/2] [UU]

unused devices: <none>
```

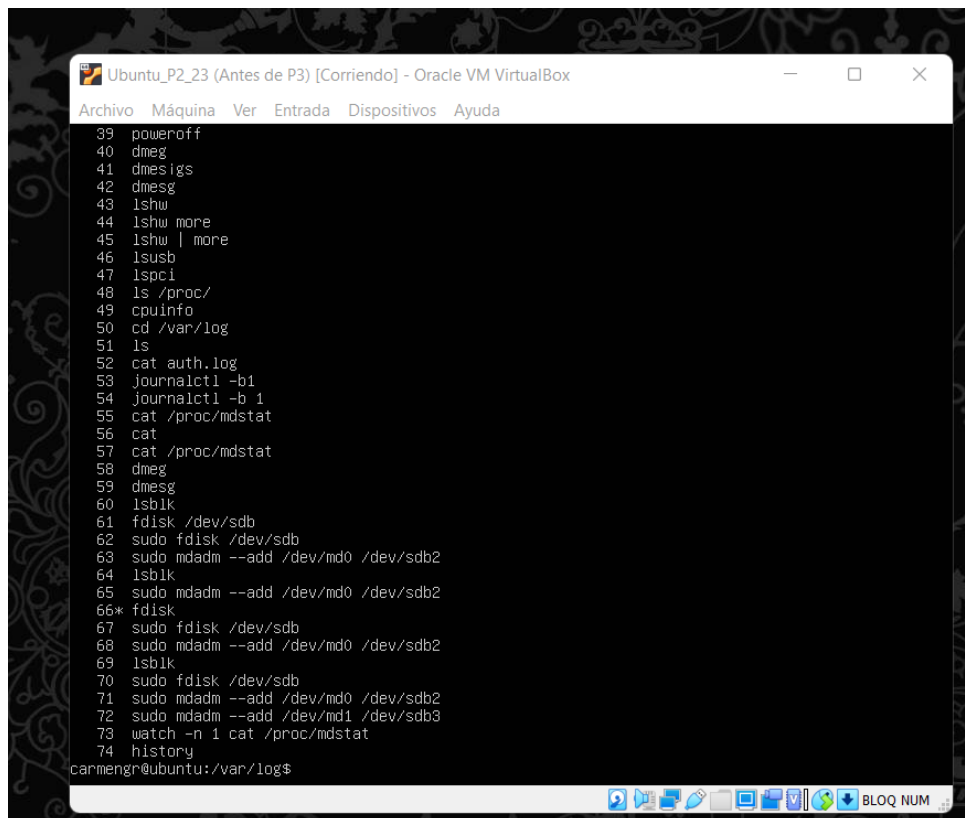
y completada



```
Ubuntu_P2_23 (Antes de P3) [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
Every 1.0s: cat /proc/mdstat                                ubuntu: Thu Nov  3 16:12:37 2022

Personalities : [raid1] [linear] [multipath] [raid0] [raid6] [raid5] [raid4] [raid10]
md1 : active raid1 sdb3[2] sda3[0]
      10063872 blocks super 1.2 [2/2] [UU]
md0 : active raid1 sdb2[2] sda2[0]
      408576 blocks super 1.2 [2/2] [UU]

unused devices: <none>
```



```
39 poweroff
40 dmesg
41 dmesigs
42 dmesg
43 lshw
44 lshw more
45 lshw | more
46 lsusb
47 lspci
48 ls /proc/
49 cpuinfo
50 cd /var/log
51 ls
52 cat auth.log
53 journalctl -b1
54 journalctl -b 1
55 cat /proc/mdstat
56 cat
57 cat /proc/mdstat
58 dmesg
59 dmesg
60 lsblk
61 fdisk /dev/sdb
62 sudo fdisk /dev/sdb
63 sudo mdadm --add /dev/md0 /dev/sdb2
64 lsblk
65 sudo mdadm --add /dev/md0 /dev/sdb2
66* fdisk
67 sudo fdisk /dev/sdb
68 sudo mdadm --add /dev/md0 /dev/sdb2
69 lsblk
70 sudo fdisk /dev/sdb
71 sudo mdadm --add /dev/md0 /dev/sdb2
72 sudo mdadm --add /dev/md1 /dev/sdb3
73 watch -n 1 cat /proc/mdstat
74 history
carmengr@ubuntu:/var/log$
```

Ahora apagamos el ordenador, vamos a ver que pasa si quitamos el disco estando el ordenador a apagado

Reiniciamos, se quedará buscando, porque no sabe si hay raid o lo q sea montado, al rato dara un error y abraira un terminal, crea un sistema de fichero en la ram → **initramfs** con unos pocos de comandos, es un minilinux para arreglar algunos problemas de linux hacemos **cat /proc/mdstat**

nos tiene que salir que algún disco tiene que salir inactivo, y hacemos

**mdadm -R /dev/md0**

**mdadm -R /dev/md1**

para arrancarlos y que salgan como activos, de esta forma podremos arrancar a mano el arrancar el RAID y hacerlo funcionar

**control D** y ya arranca

**\*\*nota:** De initramfs no hay que subir nada, solo saber que existe

## **Siguiente parte del practica**

### **Zabbix**

**Ejercicio1:** instalación de zabbix 5.0 y configurar par que se monitorice a él mismo a la máquina centos. Esta memoria se entrega el día del examen

Vamos a ver como se instala:

lo descargamos desde la página para ubuntu 20.04 queremos el server frontend y el agente y tenemos apache y la mysql

abajo nos saldrán los comandos

Hay que crear una base de datos (instalarlo y hacer mysql secure installation y configurar para poder hacer el mysql -u root -p )

Como se accede zabbix? desde el navegador ponernos la ip de nuestra máquina, nos saldrá que metamos el user y pass,

y luego de nuevo los mismos pasos para Rocky

**\*\*nota:** La segunda parte de la práctica es con ansible

**\*\*nota:** lo del mon raid es un ejercicio extra, hazlo xD

### Ejercicio2

creamos el fichero que nos dice el fichero

hacemos python script

da error porq tenemos python3, tenemos que usar python3

Creamos los otros dos ficheros que pone en el guion

Arrancamos el servicio

```
sudo systemctl enable mon RAID.timer mon RAID.service
```

```
sudo systemctl start mon RAID.timer mon RAID.service
```

**journalctl -u mon\_service --since="yesterday"**

instalamos ansible

configuramos el fichero **/etc/ansible/hosts** y añadimos las dos ip

**192.168.56.110 y 105**

Se conecta por ssh, comprobamos que funciona(ansible es un programa, no un servicio, por tanto no tenemos que reiniciarlo)

**ansible all -m ping -u carmenr** (nos conectamos a todas las máquinas de host y hacemos ping)

Nos da un error, de la conexión → , es por el puerto lo modificamos

```
sudo vi /etc/ansible/ansible.cfg → remote_port = 22022
```

**ansible all -m ping -u carmenr**

A Rocky si nos deja conectarnos pero a ubuntu no porque no hemos compartido la clave pública

**ssh carmengr@192.168.56.105 -p 22022**

**ssh-copy-id carmengr@192.168.56.105 -p 22022**

**ansible all -m ping -u carmengr**

todo ok

**ansible all -a "python3 /home/carmengr/mon RAID.py(script.py)" -u carmengr**

ubuntu sale ok script pero centos da error, porq? porque no tenemos el fichero en la maquina centos

Vamos a copiar el fichero:

**scp -p 22022 mon RAID.py carmengr@192.168.56.110:/home/carmengr/mon RAID.py**

(conectarnos por ssh y copiar un fichero de forma remota)

**ansible all -a "python3 /home/carmengr/mon RAID.py(script.py)" -u carmengr**

el resultado de esto es lo que tenemos que subir en la memoria

journalctl -u mon RAID --since="yesterday" para ver mensajes del servicio desde ayer

## Practica 3: monitorización

Apuntes clase

### Zabbix

- en la web de zabbix tenemos que buscar el apartado de instalación por comandos
- En ubuntu, en centos es más complejo
- version 5.0
- instalar primero mariadb y apache
- levantar el firewall el puerto 80 tiene que estar levantado
- en centos se instala un zabbix agent: solo atiende a peticiones del servidor que él conoce, por tanto en el fichero zabbix agent.cfg tiene una lista de ip de servidores zabbix que conoce, ahí tendremos que dar de alta nuestra ip de ubuntu.

Para probar que todo funciona: ir a la consola de zabbix y dar de alta un nuevo ordenador (host) , cuál es su ip ( el de la máquina centOs) Y NO FUNCIONARA, como depurar ? utilizando el zabbix \_get es una utilidad que conoce el protocolo de zabbix y hace peticiones. Si devuelve algo, es porq funciona correctamente. o desde centos podemos hacer un zabbix\_get a localhost.

/etc/zabbix/zabbix\_agent.cfg

pasos aprox:

1. lanzar el agente en centos
  2. lanzar una petición en local
  3. después desde ubuntu a centos
    - a. mirar si el firmware está levantado
    - b. si están conectadas las máquina para poder hacer un get
- Necesitamos ssh y http en centos para podemos monitorizar con zabbix
  - Objetivo del guión: mostrar una consola de zabbix, una shell de centos, bajar http y ver q en zabbix salta una alarma, y volver a subir http y hacer otra captura. Las señales de subida y bajada las podemos ver en las gráficas de zabbix.

Elementos de zabbix y cosas que debemos tocar:

configuration → host : para añadir una máquina

aplicar templates que te facilita la información

en monitoring --> latest data podemos ver una gráfica de las muestras tomadas

---> en dashboard veremos las alarmas

por ejemplo no va apache, ¿qué puede pasar?

- comprobar que esta instalado : **systemctl status apache**
- comprobar el firewall: **firewall-cmd list all**  
**sudo firewall-cmd --zone=public --add=http**



## Ansible

La herramienta no funciona como un servidor, corriendo en background, se ejecuta por demanda ( como un script).

Es un programa escrito en python, abrimos una shell y escribimos ansible.

Lo interesante es que en las máquinas que controlamos no es necesario instalar ansible, solo ssh y python, ( creo que ambos vienen instalados en ubuntu y centos)

En qué consiste el ejercicio:

instalar ansible en la máquina, si da muchos problema instalarlo en ubuntu q es muy rapido y vienen en un paquete. pero lo ideal es en la anfitrión

después: hacer ping en ubuntu y centos, no e sun ping de red, lo que hace es una prueba de para ver si esa lista para ejecutarse y luego hacer un power off, es decir con una sola instrucción desde el anfitrión apagar las dos maquina virtuales, dificultad: la conexión por ssh por llave pública sin introducir contraseña, una vez ahí tendremos que convertirnos en root (accediendo como un usuario normal), como convertimos en root, en ansible hay una instrucción que se llama **become**

tendremos que evitar que nos pida la contraseña de root, en el fichero /etc/sudoers (buscar en internet como quitarle a root la contraseña usando sudoers):

collection index: índice de los módulos: vamos a usar:

- el ansible bulitng ping

una vez instalado ansible en ubuntu, en /etc/ansible vendrá la configuración

more host : nos muestra como agrupar las ips

more .ansible.cfg

en etc/ansible/host **\*\*(esto es para darle nombre a nuestras máquinas)\*\***

# Ex 1: ungroupes hosts, specify before etc etc et

**ubuisse ansible\_host = 192.168.56.10 ansible\_user= carmen**

**centos ansible\_host= 192.168.56.20**

**[linux]**

**ubuisse**

**centos**

ansible ubuisse .m ping -u miNombre

para no tener ques estar poniendo nombre podemos poner en el fichero host

ubuisse ansible\_host = 192.168.56.10 **ansible\_user= carmen**

**ansible ubuisse -m ping -a 'data= "Hola ise"'** : te devuelve el mensaje de success que añadadas con -a

**ansible ubuisse -m shell -a 'ls -la'**

hace ping a las máquinas

**ansible linux -m ping** (nota: linux es la etiqueta para poder mandar comandos a lasdos máquinas

imp! : la potencia de ansible es hacer scripts que vayan haciendo tareas. ansible usa

comando idempotentes, siempre deja una tarea en un estado conocido,

nos permite ver toda la información de toda nuestra urbanización de máquinas en todo momento.

## Práctica 4\_ 23

### openBenchmarking. foronix

probar dos benchmark en ubuntu y centos, y las comentamos

**\*\*Nota: no usar benchmark de disco duro, porq son virtuales y se puede romper.**

vamos a probar ab(apache benchmark) y jmeter

ab(apache benchmark): hace peticiones a una web y vemos cómo de rápido es nuestro servidor web

**sudo apt install apache2-utils**

**ab -n 100 -c 10 <http://google.com>**

y nos da información, pero hay un problema, o podemos ver cuánto tarda google por ejemplo, en mandarle a mi usuario, loguearse y devolverme una respuesta, como modelamos esto? **con Jmeter**

jmeter lo instalando en nuestro windows y lo configuro apra que se conecte a una ip y le podemos pasar usuarios y contraseñas, un fichero de log para que se comporte como nuestro servidor.

en la segunda parte de la prácticas vamos a configurar jmeter para probar una aplicación que se ejecuta en docker.

**sudo snap install docker**

**nos vamos a descargar la aplicacion del profesor David de git**

**git clone <https://direccion> que sal en el guion**

nos metemos en el directorio y hacemos

**sudo docker-compose up**

se va a bajar la imagen de mongo y node.js

las levantamos y podremos conectarnos desde la api en el navegador, e introduciendo parámetros podremos obtener información.

Nos dará un error, es porque tenemos que loguearnos

Hay que hacer un post con el user y pass para acceder a la api, después enviamos el usr y pass de un user en concreto, nos devuelve un numero , ese lo utilizamos para hacer una petición para consultar infor de un alumno, poniendo en la cabecera “ Authorization: Bearer \$token”

### **Segunda parte**

Configurar jmeter, lo ejecutamos y en la parte izquierda tendremos las configuraciones que tenemos que hacer

## **Práctica 4**

Apuntes clase

descargar el proyecto desde github con un clone  
instalar docker en la máquina en la que vamos a ejecutar

vamos a usar los verbos de http. Los mas comunes con

- get (consulta)
- post-put (inserción, modificar)
- patch
- delete

Solo vamos a poder consultar el expediente de una alumno

GET /api/vi/alumnos/aulas/id : para lectura

esta api ofrece dos endpoint

- auth/login : para identificación
- /alumnos/alumno

el servicio va a correr sobre una de nuestras maquinas y nuestro ordenador anfitrión vamos a correr jmeter y vamos a usar la red para atacar los dos endpoint

**Nota:**

al instalar docker, tendremos que hacerlo con root o si queremos hacerlo desde nuestro usuario para evitar hacer sudo todo el rato → post-installation steps for linux ahi encontraremos los comandos para hacerlo

Dentro de la aplicación tenemos dos contenedores:

- nodejs pone la api
- otro mas efímero de mongodb que lo unico que hace es inicializar la base de datos con unos pocos de datos para usar

estos contenedores los corre docker, **para que docker compose entonces?** es un orquestador de contenedores, lo que hace es levantar los servicios (mongodb, init mongodb y nodejs, y crea una red virtual entre ellos para que se puedan comunicar entre ellos

Una vez instalados docker y descargado el proyecto.

**docker-compose up** con esto lo arrancamos

arrancará los servicios y podremos ver los logs, para hacer la depuración es más sencillos pero se mezclan los log de node y mongo. Si no queremos podemos hacer docker-compose up -d

si queremos ver los logs docker-compose up ....

una vez que accedemos a la página principal del proyecto, nos muestra información y nos devuelve un token jwt ( como una cookie, pero la filosofía de uso es diferente porq es el usuario el que tiene que mantener su informacion ahi y dársela al servidor) obtendremos un script pruebaEntorno.sh tiene información y dos llamadas.

basicAuth protocolo de autenticación antiguo ya apenas no se usa

lo usamos para evitar ruido de internet, el propósito no es identificar al usuario que hace la llamada si no que para q os clientes que consumen la api sepamos que son autorizados

hacemos ./pruebaEntonces.sh

obtendremos un token y el expediente

en la web de documentación jwt hay un depurador de token, podemos copiar nuestro token y ver la información obtenida

### **Seguimos:**

En el navegador, en la web jsonpretty podemos copiar lo obtenido como expediente y vemos la información mejor presentada.

Ahora en alumnos/alumno

tenemos dos tipos de usuarios alumnos y admins, de forma que un alumno puede obtener un expediente y un admin puede obtener grupos de expedientes

En el proyecto hay ficheros con información de alumnos ficticios, que vamos usar para simular cargas. Como alumnos vamos a simular las cargas y como administradores vamos a usar los get de apache.

usuario tendremos http log access

acceso log samples para adminis



## LECCION 2: Centos

el cliente quiere videos de alta calidad y larga duración , para soportar esta carga vamos a utilizar el /var y ubicarlo dentro de un volumen lógico distinto y añadirle un disco extra

### Pasos:

añadimos el disco virtual, configuramos el so, creamos un usuario lo hacemos admin, y ponemos contraseña al root y usuario (carmen, practicarse), dejamos el disco duro que le asigna por defecto.. Cuando esto este hecho, reiniciamos.

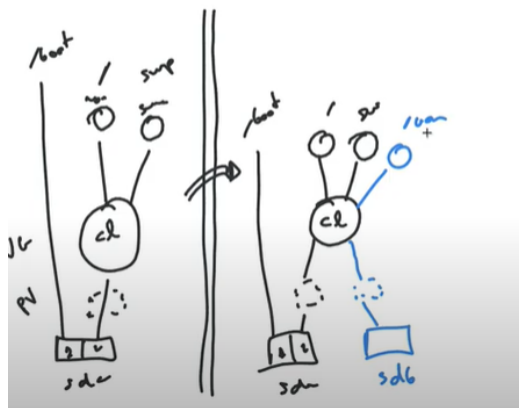
Eliminamos el disco óptico desde la configuración y volvemos a reiniciar.

Introducimos nuestro usuario y contraseña y ejecutamos el comando lsblk para ver las particiones ya creadas. CentOS por defecto crea el disco sda y una partición de un G donde tiene /boot asignado como punto de montaje y luego tenemos una partición sda2 de los 7 g restantes del disco, donde tenemos gestionado por lvm un grupo de volúmenes, denominados cl, q tiene dos volúmenes lógicos, el root y swap, donde le asigna 820 y 6,2 g respectivamente.

### Representación gráfica

De esta configuración inicial que viene por defecto. Tenemos que pasar a la siguiente configuración

añadiendo sdb, creando su fv, añadirlo a cl y crear un nuevo volumen lógico para asignarle /var



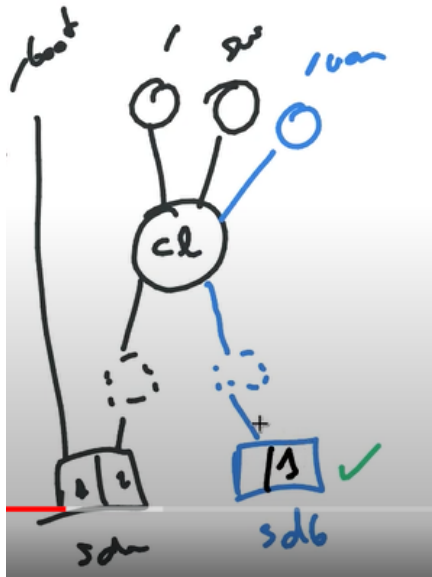
Apagamos la maquina y añadimos un disco, comprobamos que se ha creado el disco(q la maquina lo ha detectado) con lsblk ( lsblk | grep disk, para verlo para claro) y vemos que tenemos sda y sdb

Sera recomendable crear un particion para tener los metadatos

sudo fdisk/dev/sdb entramos al menu y podemos darle a m para ver las distintas opciones, con p las particiones, vemos que no hay ninguna y creamos una nueva.

le damos n, para crearla, p para q sea primaria, 1 el numero de particion, y vemos q le deja unos 2megas de espacio al principio par ametadatos y group y aprovecha el max del disco, ya cuando e damos p vemos que tenemos sdb1 y con w pasamos a escribirla.

Ahora tecleando lsblk tenemos sdb y la partición sdb1. Entonces en nuestro esquema tenemos:



Ahora vamos a pasar a crear nuestro physical volume. Para interactuar con lvm:

1. sudo lvm donde presionamos tab dos veces y vemos todos los comandos o bien
2. podemos invocar los comandos desde el terminal directamente sin tener la mini consola.

Por ejemplo para obtener la info sobre los fisical volumen tenemos pvdisplay (siempre como superusuario. a modo reducido tenemos sudo pvs. El siguiente paso es crear nuestro physical volumen a partir del dispositivo sdb1.

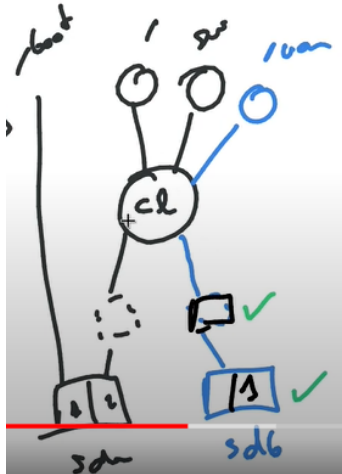
**sudo pvcreate**, indicando el dispositivo a partir del cual vamos a crear este physical volumen.

### sudo pvcreate /dev/sdb1

se crea de manera exitosa, nuestro physical volumen que coge todo el tam en este caso los 8GB. Vamos a comprobarlo: **sudo pvs** y **sudo pvdisplay**

```
[carmen@localhost ~]$ sudo pvcreate /dev/sdb1
[sudo] password for carmen:
Physical volume "/dev/sdb1" successfully created.
[carmen@localhost ~]$ sudo pvs
  PU      VG Fmt Attr PSize  PFree
/dev/sda2 cl  lvm2 a--  <7,00g    0
/dev/sdb1 lvm2 ---  <8,00g  <8,00g
[carmen@localhost ~]$ sudo pvdisplay
--- Physical volume ---
PU Name      /dev/sda2
VG Name      cl
PU Size      <7,00 GiB / not usable 3,00 MiB
Allocatable  yes (but full)
PE Size      4,00 MiB
Total PE     1791
Free PE      0
Allocated PE 1791
PU UUID      7RtJXR-sJUT-g3EE-hZsN-gCb1-C8BE-d7F7f6

"/dev/sdb1" is a new physical volume of "<8,00 GiB"
--- NEW Physical volume ---
PU Name      /dev/sdb1
VG Name
PU Size      <8,00 GiB
Allocatable  NO
PE Size      0
Total PE     0
Free PE      0
Allocated PE 0
PU UUID      ZzdQ7b-ARk8-KqZ5-nben-CjGB-JorS-mapdwG
[carmen@localhost ~]$
```



Ya tenemos el physical volumen `/dev/sdb1`, listo y ahora tenemos que extender nuestro volumen group para que coja este physical volumen, para monitorizar el estado de nuestro grupo de volúmenes **sudo vgdisplay** o **vgs** (para lista corta)

y el comando para ver los posibles comandos: **sudo vg**

Para extender el tamaño del grupo de volúmenes tenemos **vgextend**

Tenemos que indicar el grupo de volúmenes y los physical volumen, **vgextend VG PV...**

(Es interesante recordar ciertos comandos para detectar errores)

En este caso vamos a extender `cl` con el physical volumen `/dev/sdb1`:

**sudo vgextend cl /dev/sdb1**

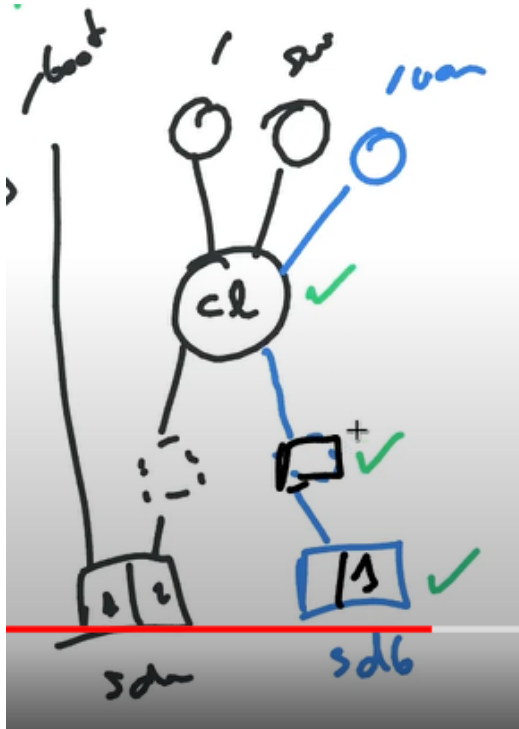
y ahora tecleamos **sudo vgs** para ver que se ha hecho correctamente

```
[carmen@localhost ~]$ sudo vgextend cl /dev/sdb1
Volume group "cl" successfully extended
[carmen@localhost ~]$ sudo vgs
VG #PV #LV #SN Attr   USize  UFree
cl  2  2  0 wz--n- 14,99g <8,00g
[carmen@localhost ~]$
```

Con **vgdisplay** también veremos que tenemos dos PV y dos LV, y que se ha aumentado el la capacidad del VG y el espacio que tenía libre.

Nuestro esquema: ya tenemos nuestro `cl` extendido con nuestro nuevo physical volumen, ya solo queda crear nuestro logical volumen.





Para crear el logical volume, primero tenemos **sudo lvs** para ver la información de los volúmenes que tenemos y con **lvdisplay**, información un poco mas detallada.

Tenemos **sudo lvcreate**, donde tenemos varios parámetros. Con **lvcreate** podemos tambien crear una snapshot a partir de un logical volúmenes. También tenemos **lvm** para configurar raid 1. etc

En este caso vamos a asignarle un nombre e indicar la longitud, tenemos 8 GB podemos usar 6 y luego vemos si redimensionar o no ( en el vio de tiene 4G y usa 3, pero yo al principio deje el disco inicial con 8G) y por ultimo especificar el grupo de volúmenes donde queremos crear ese volumen logico, puesto que en esta configuración tenemos solo **cl**, pero podriamos tener mas grupos:

**sudo lvcreate -n new\_var -L 3G cl**

```
[carmen@localhost ~]$ sudo lvcreate -n new_var -L 3G cl
[sudo] password for carmen:
  Logical volume "new_var" created.
[carmen@localhost ~]$ _
```

## Comprobamos

```
--- Logical volume ---
LV Path                /dev/cl/root
LV Name                root
VG Name                cl
LV UUID                jYwCxb-yUY2-I1tA-Gcp9-VRBX-5w0N-sAxZxU
LV Write Access        read/write
LV Creation host, time localhost, 2021-10-08 10:47:07 -0400
LV Status              available
# open                 1
LV Size                <6,20 GiB
Current LE             1586
Segments               1
Allocation             inherit
Read ahead sectors     auto
  - currently set to   8192
Block device           253:0

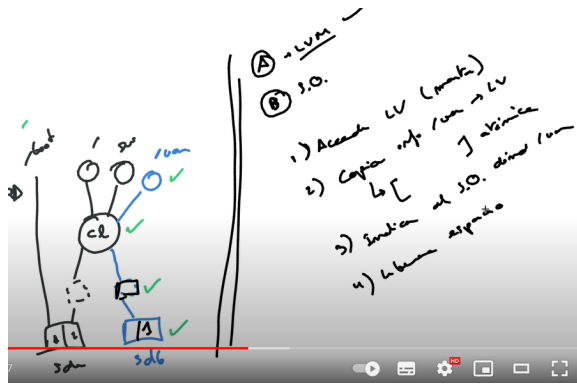
--- Logical volume ---
LV Path                /dev/cl/new_var
LV Name                new_var
VG Name                cl
LV UUID                QgtEbp-bbeU-96tC-f7Kd-pZBL-vkpc-ZYJ34w
LV Write Access        read/write
LV Creation host, time localhost.localdomain, 2021-10-22 05:14:36 -0400
LV Status              available
# open                 0
LV Size                3,00 GiB
Current LE             768
Segments               1
Allocation             inherit
Read ahead sectors     auto
  - currently set to   8192
Block device           253:2

lcarmen@localhost ~]$
```

```
[lcarmen@localhost ~]$ sudo lvs
LV      VG Attr      LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
new_var cl -wi-a----- 3,00g
root    cl -wi-ao----- <6,20g
swap    cl -wi-ao----- 820,00m
[lcarmen@localhost ~]$
```

Entonces , a nivel de configuración lo que es la parte de gestión de lvm ( la parte A del procedimiento, y estaría lista). Ahora quedan cuestiones relativas a SOs que vamos a repasar igualmente.

- En primer lugar tenemos que acceder al LV (montarlo), para que el SO pueda acceder a el,
2. Copiaremos la información de /var/ al LV, esta copia la haremos de forma atómica.
  3. Tenemos que indicar al SO donde va /var
  4. Por último, liberar espacio



## Montar el VL

Vamos a asignar un punto de montaje, lo creamos:

```
sudo mkdir /new_var
```

y ahora ahora vamos a usar el comando mount ( revisamos el man, man mount)

vemos en el man

## NAME

## mount - mount a filesystem

Que sistema de archivo tiene el volumen lógico que acabamos de crear? no lo sabemos porq no lo hemos creado, el paso 0 debe ser crear un filesystema para el LV y luego ya lo montamos.

Para crear ese filesystem ( manual, man mkfs). Tenemos que especificar el tipo y el lugar donde lo vamos a crear. En nuestro caso , el volumen lógico

Tenemos distintas opciones, por tanto tenemos que pensar cual usar y los parametros.

Tenemos que coger un sistema de archivos adecuados para esos grandes archivos que tenemos, entonces de los que vemos el más adecuado es xfs, aunque es cierto que para redimensionar lo vi etc a día de hoy el que tiene la implementación más robusta en centos, en red hat si existe la opción de redimensionar hacia atrás (hacer más pequeño un filesystem xfs) pero en la versión que tenemos de CentOS 7, así que vamos a crearlo del tipo ext4:

```
sudo mkfs -t ext4 /dev/cl/new var
```

```
[carmen@localhost ~]$ sudo mkfs -t ext4 /dev/cl/new_var
[sudol] password for carmen:
mke2fs 1.45.4 (23-Sep-2019)
Se está creando un sistema de ficheros con 786432 bloques de 4k y 196608 nodos-i
UUID del sistema de ficheros: ad06a14f-2d4c-446b-ac25-5972aa4f3698
Respaldos del superbloque guardados en los bloques:
    32768, 98304, 163840, 229376, 294912

Reservando las tablas de grupo: hecho
Escribiendo las tablas de nodos-i: hecho
Creando el fichero de transacciones (16384 bloques): hecho
Escribiendo superbloques y la información contable del sistema de ficheros: hecho

[carmen@localhost ~]$
```

Ya que tenemos el sistema de archivos creados, vamos a hacer accesible mediante mount y le vamos a decir que monte nuestro dispositivo.

```
[carmen@localhost ~]$ mount /dev/mapper/cl-  
mount: /dev/mapper/cl-: No such file or directory  
[carmen@localhost ~]$ mount /dev/mapper/cl-  
cl-new_var cl-root cl-swap  
[carmen@localhost ~]$ sudo mount /dev/new_var /new_var  
[sudo] password for carmen:  
mount: /new_var: el dispositivo especial /dev/new_var no existe.  
[carmen@localhost ~]$ sudo mount /dev/cl/new_var /new_var  
[ 4776.150118] EXT4-fs (dm-2): mounted filesystem with ordered data mode. Opts: (null)  
[carmen@localhost ~]$
```

**sudo mount /dev/cl/new\_var /new\_var/**

Tecleando mount vemos que el volumen lógico está montado sobre el directorio new-var

```
proto=5,direct,pipe_ino=17855)  
mqueue on /dev/mqueue type mqueue (rw,relatime,seclabel)  
debugfs on /sys/kernel/debug type debugfs (rw,relatime,seclabel)  
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,seclabel,pagesize=2M)  
/dev/sda1 on /boot type ext4 (rw,relatime,seclabel)  
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,seclabel,size=83844k,mode=700,uid=1000,gid=1000)  
/dev/mapper/cl-new_var on /new_var type ext4 (rw,relatime,seclabel)  
[carmen@localhost ~]$ _
```

Ahora pasamos a hacer la copia de los datos. Cuando decimos que es atómica, es que cuando estamos haciendo una copia y vamos avanzando alguien modifique a posteriori mientras no hemos acabado de copiar todo, para evitar eso podemos tomar una snapshot y requerir espacio suficiente para tener ese snapshot y otra posibilidad es evitar que alguien haga esa escritura. Vamos a coger esa segunda opción por seguridad y evitaremos que algún proceso no necesario para el sistema y que los usuarios se salgan, en el caso de un servidor pues avisar de que se va a hacer un mantenimiento, por lo que no podrán acceder.

Comando **systemctl**

Controla systemd un trozo de software que rompe un poco la filosofía de unix de hacer una única cosa pero muy bien controla los servicios lo monitoriza etc etc

Red hat lo tiene, es un software bastante completo

**sudo systemctl isolate rescue**

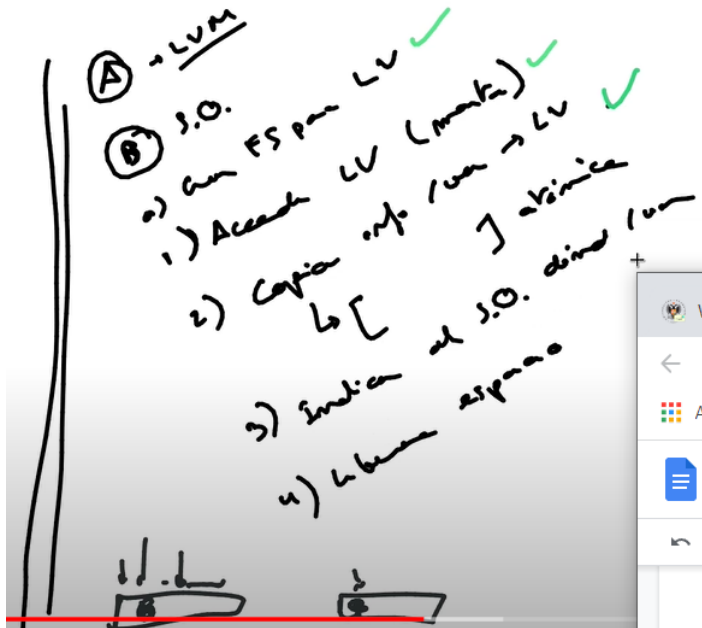
Hay un pequeño bug así que nos sacará del usuario. Volvemos a entrar como root, hacemos **systemctl status** vemos que seguimos en running, volvemos a hacer **sudo systemctl isolate rescue** y comprobamos el status de nuevo y ya vemos que si estamos en mantenimiento

**Vamos a hacer la copia, para preservar todos los atributos y metadatos vamos a usar cp con parámetros, hacemos**

**cp -a /var/. /new\_var/**

y hacemos **ls -laZ /var/** para ver que las cosas siguen ahí el -Z es para ver el contexto

Todo bien



Una vez que ya hemos copia la info lo que tenemos q decirle al So es donde asignar el punto de montaje /var (

Para ello debemos irnos al archivo del filesystem table (fstab), lo tenemos en /etc/  
hacemos `cd /etc/` y hacemos `ls`  
Podemos encontrar ayuda sobre el archivo `man fstab`  
vamos a editarlo con `vi`  
**`vi /etc/fstab`**

**añadimos `/dev/mapper/cl-new_var /var ext4 defaults 0 0`**  
**los dos enteros corresponde a la prioridad en caso de back up**

ins para entrar en modo edición, escape para salir y `:wq` para guardar los cambios

Ahora si hacemos `mount` vemos que `new_var` esta montado en el directorio `new_var`  
Asique hacemos **`umount /new_var`** y ahora haciendo `mount -a` va a montar todos los archivos dentro del filesystem table.

Ejecutamos `lsblk` para ver donde tenemos /var

y hacemos `mount -a` y `lsblk` para ver como tenemos el /var montado en `new_var`

```
[root@localhost etc]# umount /new_var
[root@localhost etc]# lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
┌sda
├sda1                8:1   0    1G 0 part /boot
├sda2                8:2   0    7G 0 part
└┬c1-root            253:0   0  6.2G 0 lvm /
  └c1-swap           253:1   0  820M 0 lvm [SWAP]
sdb                  8:16   0    8G 0 disk
└sdb1                8:17   0    8G 0 part
  └c1-new_var        253:2   0    3G 0 lvm /var
sr0                  11:0    1 1024M 0 rom

[root@localhost etc]# mount -a
[ 1683.013759] EXT4-fs (dm-2): mounted filesystem with ordered data mode. Opts: (null)
[root@localhost etc]# lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
┌sda
├sda1                8:1   0    1G 0 part /boot
├sda2                8:2   0    7G 0 part
└┬c1-root            253:0   0  6.2G 0 lvm /
  └c1-swap           253:1   0  820M 0 lvm [SWAP]
sdb                  8:16   0    8G 0 disk
└sdb1                8:17   0    8G 0 part
  └c1-new_var        253:2   0    3G 0 lvm /var
sr0                  11:0    1 1024M 0 rom

[root@localhost etc]# _
```

Vemos que tenemos /var montado en nuestro nuevo volumen logico /new\_var

Ahora lo que queda es liberar espacio, para ello tenemos q ir marcha atras:

- **editamos el fstab: comentado la linea previamente añadida( # /dev/mapper...**
- **hacemos mount -a**
- **umount /dev/cl/new\_var**

```
CentOS8 [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda

"fstab" 15L, 629C written
[root@localhost etc]# mount -a
[root@localhost etc]# lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
┌sda
├sda1                8:1   0    1G 0 part /boot
├sda2                8:2   0    7G 0 part
└┬c1-root            253:0   0  6.2G 0 lvm /
  └c1-swap           253:1   0  820M 0 lvm [SWAP]
sdb                  8:16   0    8G 0 disk
└sdb1                8:17   0    8G 0 part
  └c1-new_var        253:2   0    3G 0 lvm /var
sr0                  11:0    1 1024M 0 rom

[root@localhost etc]# umount /dev/cl/new_var
[root@localhost etc]# lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
┌sda
├sda1                8:1   0    1G 0 part /boot
├sda2                8:2   0    7G 0 part
└┬c1-root            253:0   0  6.2G 0 lvm /
  └c1-swap           253:1   0  820M 0 lvm [SWAP]
sdb                  8:16   0    8G 0 disk
└sdb1                8:17   0    8G 0 part
└c1-new_var          253:2   0    3G 0 lvm
sr0                  11:0    1 1024M 0 rom

[root@localhost etc]#
```

Ahora volvemos a editar el fstab para asegurarnos de que cuando lo llamemos está preparado para funcionar, en caso de que haya que reiniciar la máquina: descomentamos la linea de nuevo.

Y ahora vamos a mover /var a /var\_old con la finalidad de que si hay algún problema podemos recuperar la información a golpe de mover un directorio.

```
mv /var /var_old
```

y volvemos a hacer el mount -a

**Q ocurre?** Q como hemos movido /var a /var\_old ahor amismo cuando llamamos a mount-a nos dice que el punto de montaje no existe, por tanto hay que crearlo a mano: **mkdir /var**

y ahora si hacemos **ls / -laZ** tenemos que el contexto de ese linu para vardifiere del var old, para ello tenemos el ecomanto **restorecon**, que restaura el contexto de los directorios correspondientes , se lo aplicamos a /var : **restorecon /var**

```
dr-xr-xr-x. 84 root root system_u:object_r:proc_t:s0      0 nov  3 05:15 proc
dr-xr-xr-x.  2 root root system_u:object_r:admin_home_t:s0 135 oct 22 06:07 root
drwxr-xr-x. 28 root root system_u:object_r:var_run_t:s0    720 nov  3 05:20 run
lrwxrwxrwx.  1 root root system_u:object_r:bin_t:s0        8 may 10 2019 sbin -> usr/sbin
drwxr-xr-x.  2 root root system_u:object_r:var_t:s0        6 may 10 2019 srv
dr-xr-xr-x. 13 root root system_u:object_r:sysfs_t:s0      0 nov  3 05:15 sys
drwxrwxrwt.  7 root root system_u:object_r:tmp_t:s0        93 nov  3 05:33 tmp
drwxr-xr-x. 12 root root system_u:object_r:usr_t:s0       144 oct  8 10:47 usr
drwxr-xr-x.  2 root root unconfined_u:object_r:default_t:s0 6 nov  3 05:58 var
drwxr-xr-x. 21 root root system_u:object_r:var_t:s0       4096 oct  8 10:52 var_old
[root@localhost etc]# restorecon /var
[root@localhost etc]# ls / -laZ
total 24
dr-xr-xr-x. 19 root root system_u:object_r:root_t:s0      254 nov  3 05:58 .
dr-xr-xr-x. 19 root root system_u:object_r:root_t:s0      254 nov  3 05:58 ..
lrwxrwxrwx.  1 root root system_u:object_r:bin_t:s0        7 may 10 2019 bin -> usr/bin
dr-xr-xr-x.  6 root root system_u:object_r:boot_t:s0      4096 oct  8 10:52 boot
drwxr-xr-x. 20 root root system_u:object_r:device_t:s0     3160 nov  3 05:15 dev
drwxr-xr-x. 82 root root system_u:object_r:etc_t:s0       8192 nov  3 05:55 etc
drwxr-xr-x.  3 root root system_u:object_r:home_root_t:s0  20 oct  8 10:50 home
lrwxrwxrwx.  1 root root system_u:object_r:lib_t:s0        7 may 10 2019 lib -> usr/lib
lrwxrwxrwx.  1 root root system_u:object_r:lib_t:s0        9 may 10 2019 lib64 -> usr/lib64
drwxr-xr-x.  2 root root system_u:object_r:mnt_t:s0        6 may 10 2019 media
drwxr-xr-x.  2 root root system_u:object_r:mnt_t:s0        6 may 10 2019 mnt
drwxr-xr-x. 21 root root system_u:object_r:var_t:s0       4096 oct  8 10:52 new_var
drwxr-xr-x.  2 root root system_u:object_r:usr_t:s0        6 may 10 2019 opt
dr-xr-xr-x. 84 root root system_u:object_r:proc_t:s0      0 nov  3 05:15 proc
dr-xr-xr-x.  2 root root system_u:object_r:admin_home_t:s0 135 oct 22 06:07 root
drwxr-xr-x. 28 root root system_u:object_r:var_run_t:s0    720 nov  3 05:20 run
lrwxrwxrwx.  1 root root system_u:object_r:bin_t:s0        8 may 10 2019 sbin -> usr/sbin
drwxr-xr-x.  2 root root system_u:object_r:var_t:s0        6 may 10 2019 srv
dr-xr-xr-x. 13 root root system_u:object_r:sysfs_t:s0      0 nov  3 05:15 sys
drwxrwxrwt.  7 root root system_u:object_r:tmp_t:s0       93 nov  3 05:33 tmp
drwxr-xr-x. 12 root root system_u:object_r:usr_t:s0       144 oct  8 10:47 usr
drwxr-xr-x.  2 root root unconfined_u:object_r:var_t:s0    6 nov  3 05:58 var
drwxr-xr-x. 21 root root system_u:object_r:var_t:s0       4096 oct  8 10:52 var_old
[root@localhost etc]#
```

Ya ha cambiado el contexto a /var

Ahora hacemos mount -a y lsblk, tenemos nuestro var montado y si hacemos **ls /** vemos que tenemos esos bits accesibles a través del directorio en /var\_old.

```
[root@localhost etc]# mount -a
[ 2860.815619] EXT4-fs (dm-2): mounted filesystem with ordered data mode. Opts: (null)
[root@localhost etc]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda           8:0    0   8G  0 disk
├─sda1        8:1    0   1G  0 part /boot
├─sda2        8:2    0    7G  0 part
│   └─cl-root 253:0    0  6.2G  0 lvm  /
│       └─cl-swap 253:1    0 820M  0 lvm  [SWAP]
sdb           8:16    0   8G  0 disk
├─sdb1        8:17    0   8G  0 part
│   └─cl-new_var 253:2    0    3G  0 lvm  /var
sr0          11:0    1 1024M  0 rom

[root@localhost etc]# ls /
bin  dev  home  lib64  mnt      opt  root  sbin  sys  usr  var_old
boot  etc  lib  media  new_var  proc  run  srv  tmp  var
```

Y con esto termina la práctica

**exit y poweroff**

## Práctica 1. Lección 3:

Tras ver el éxito de los vídeos alojados en el servidor configurado en la práctica anterior, un amigo de su cliente quiere proceder del mismo modo pero va a necesitar alojar información sensible así que le pide explícitamente que cifre la información y que ésta esté siempre disponible. Por tanto, la decisión que toma es configurar un RAID1 por software y cifrar el VL en el que /var estará alojado.

Deseamos que /var vaya a un raid 1 cifrado, entonces vamos a tener nuestro sda particionado en /boot y un physical volume sus dos logical volumen, y queremos un /var nuevo. Pues ya sabemos que podemos añadir nuestro disco sdb y extenderlo, pero que problema, que no estaría cifrado entonces vamos a tener que añadir como mínimo dos discos y a partir de ahí crear el md0. Sobre eset multidevice configurado como RAID1 crearemos el fiscal volumen y ahora tenemos dos opciones:

- extender el grupo de volumen con nuestro nuevo fiscal volumen y a partir de ahí crear el logical volume, qué ocurre? que no tendríamos la certeza de que /var vaya a parar a esta configuración. En este caso concreto sabemos que sí ocurriría porque sda está completa y la nueva información va a sdb, pero podría no ser el caso y no tener esa certeza.

Entonces la solución de diseño más correcta pasa por crear a partir de este nuevo physical volume un nuevo volumen group y a partir de este podemos meter ahí todos los volúmenes que queremos que vayan en la configuración de raid 1.

Podríamos tener el caso también de que nos interesa por prestación tener los logical volumen en unos dispositivos más rápidos, entonces seríamos un grupo de volúmenes con esos dispositivos para eso logical volumen que necesitan más velocidad.



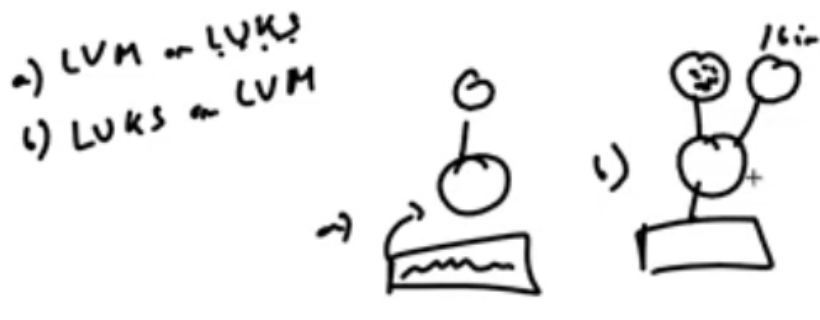
Además nos piden que /var este cifrado entonces vamos a cifrar esta información, y veremos que se nos presentan dos posibilidades:

- **LVM sobre LUKS**
- **LUKS sobre LVM**

En el primer caso lo q vamos a tener es un cifrado de disco a partir de ahí se cifra todo el disco y una vez activa es formateado, se crea el volumen group y el logical volume

en el esquema b) escoger utilizar LVM y luego el logical volume es lo que se cifra





Nos pasamos a la máquina virtual.

Primero vamos a añadir los dos discos, sdb y sdc, de 2GB

Iniciamos la máquina y, como siempre, primero comprobamos que nuestra máquina tiene los dos discos con lsblk.

Ahora vamos a crear el RAID 1, con la novedad de esta práctica, el comando mdadm (multi device administration) es un comando bastante complejo porque tiene una gran cantidad de opciones.

Primero vamos a crear las particiones en sdb y sdc con fdisk. Formateamos:

**sudo fdisk /dev/sdb**

con p, vemos primero que la partición no está.

```

lcarmen@localhost ~1$ sudo fdisk /dev/sdb

Bienvenido a fdisk (util-linux 2.32.1).
Los cambios solo permanecerán en la memoria, hasta que decida escribirlos.
Tenga cuidado antes de utilizar la orden de escritura.

El dispositivo no contiene una tabla de particiones reconocida.
Se ha creado una nueva etiqueta de disco DOS con el identificador de disco 0xb056fecf.

Orden (m para obtener ayuda): p
Disco /dev/sdb: 2 GiB, 2147483648 bytes, 4194304 sectores
Unidades: sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico/físico): 512 bytes / 512 bytes
Tamaño de E/S (mínimo/óptimo): 512 bytes / 512 bytes
Tipo de etiqueta de disco: dos
Identificador del disco: 0xb056fecf

Orden (m para obtener ayuda): _
  
```

Con n creamos una nueva, y le dejamos el valor por defecto

```

Orden (m para obtener ayuda): n
Tipo de partición
  p  primaria (0 primaria(s), 0 extendida(s), 4 libre(s))
  e  extendida (contenedor para particiones lógicas)
Seleccionar (valor predeterminado p): p
Número de partición (1-4, valor predeterminado 1):
Primer sector (2048-4194303, valor predeterminado 2048):
Último sector, +sectores o +tamaño{K,M,G,T,P} (2048-4194303, valor predeterminado 4194303):

Crea una nueva partición 1 de tipo 'Linux' y de tamaño 2 GiB.

Orden (m para obtener ayuda): _
  
```

Con **p** comprobamos que esta correcto:

```
Orden (m para obtener ayuda): p
Disco /dev/sdb: 2 GiB, 2147483648 bytes, 4194304 sectores
Unidades: sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico/físico): 512 bytes / 512 bytes
Tamaño de E/S (mínimo/óptimo): 512 bytes / 512 bytes
Tipo de etiqueta de disco: dos
Identificador del disco: 0xb856fecf

Disposit.  Inicio Comienzo  Final Sectores Tamaño Id Tipo
/dev/sdb1      2048 4194303  4192256      2G 83 Linux

Orden (m para obtener ayuda): _
```

Con **w** salimos

Hacemos lo mismo para **sdc**

```
[carmen@localhost ~]$ sudo fdisk /dev/sdc

Bienvenido a fdisk (util-linux 2.32.1).
Los cambios solo permanecerán en la memoria, hasta que decida escribirlos.
Tenga cuidado antes de utilizar la orden de escritura.

El dispositivo no contiene una tabla de particiones reconocida.
Se ha creado una nueva etiqueta de disco DOS con el identificador de disco 0xd2f6c52e.

Orden (m para obtener ayuda): n
Tipo de partición
  p primaria (0 primaria(s), 0 extendida(s), 4 libre(s))
  e extendida (contenedor para particiones lógicas)
Seleccionar (valor predeterminado p): p
Número de partición (1-4, valor predeterminado 1):
Primer sector (2048-4194303, valor predeterminado 2048):
Último sector, +sectores o +tamaño{K,M,G,T,P} (2048-4194303, valor predeterminado 4194303):

Crea una nueva partición 1 de tipo 'Linux' y de tamaño 2 GiB.

Orden (m para obtener ayuda): p
Disco /dev/sdc: 2 GiB, 2147483648 bytes, 4194304 sectores
Unidades: sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico/físico): 512 bytes / 512 bytes
Tamaño de E/S (mínimo/óptimo): 512 bytes / 512 bytes
Tipo de etiqueta de disco: dos
Identificador del disco: 0xd2f6c52e

Disposit.  Inicio Comienzo  Final Sectores Tamaño Id Tipo
/dev/sdc1      2048 4194303  4192256      2G 83 Linux

Orden (m para obtener ayuda): _
```

Con **w** salimos

Y comprobamos que efectivamente se han hecho las particiones, con **lsblk**:

```
[carmen@localhost ~]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda          8:0    0   8G  0 disk
├─sda1       8:1    0    1G  0 part /boot
├─sda2       8:2    0    7G  0 part
│   ├─cl-root 253:0    0  6,2G  0 lvm  /
│   └─cl-swap 253:1    0 820M  0 lvm  [SWAP]
sdb          8:16    0    2G  0 disk
└─sdb1       8:17    0    2G  0 part
sdc          8:32    0    2G  0 disk
└─sdc1       8:33    0    2G  0 part
sr0         11:0    1 1024M  0 rom
```

Ahora vamos a crear nuestro RAID 1, para definirlo tenemos que invocar el comando **mdadm**, especificar el modo en el que estamos operando, el dispositivo con el que queremos operar y las distintas opciones y en último lugar ubicamos los distintos dispositivos.

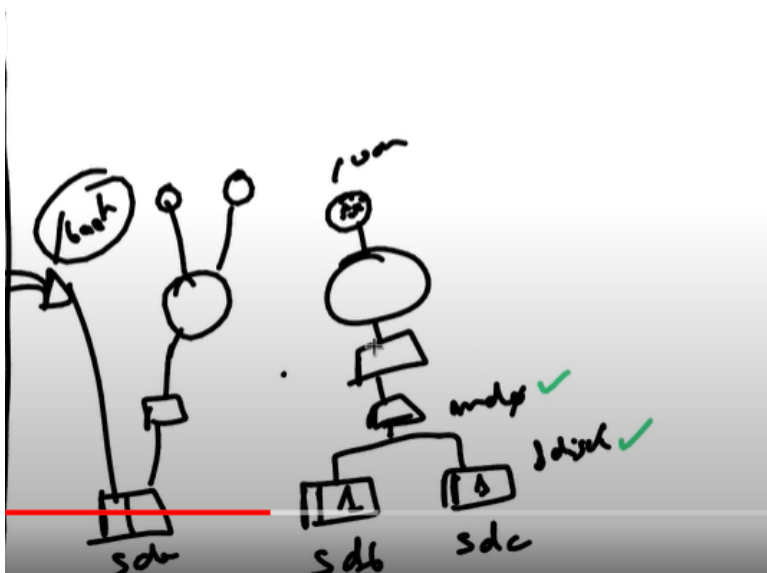
Comando, como superusuario:

**sudo mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1**  
 indicamos que estamos en nivel 1 y que vamos a usar dos dispositivos  
 Nos indica que no vamos a dejar espacio para los metadatos pero en este caso nos da igual, así que continuamos, y

```
[carmen@localhost ~]$ sudo mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1
[sudo] password for carmen:
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
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
[ 1144.002842] md/raid1:md0: not clean -- starting background reconstruction
[ 1144.003240] md/raid1:md0: active with 2 out of 2 mirrors
[ 1144.003604] md0: detected capacity change from 0 to 2144337920
mdadm: array /dev/md0 started.
[ 1144.005289] md: resync of RAID array md0
[carmen@localhost ~]$
```

Haciendo **ls /dev/** veremos nuestro md0

```
[carmen@localhost ~]$ [ 1154.518438] md: md0: resync done.
ls /dev
autofs          fuse            null            sg1             tty15           tty31           tty48           tty7            vcs5
block           hpjet           nvram           sg2             tty16           tty32           tty49           tty8            vcs6
bsg             hugepages       port            sg3             tty17           tty33           tty5             tty9            vcsa
bus             hwang           ppp             shm             tty18           tty34           tty50           ttyS0           vcsa1
cdrom           initctl         ptmx           snapshot        tty19           tty35           tty51           ttyS1           vcsa2
char            input           pts            snd             tty2            tty36           tty52           ttyS2           vcsa3
cl              kmsg           random          sr0            tty20           tty37           tty53           ttyS3           vcsa4
console         log            raw            stderr          tty21           tty38           tty54           uhid            vcsa5
core            loop-control    rtc            stdin           tty22           tty39           tty55           uinput          vcsa6
cpu            mapper          rtc0           stdout          tty23           tty4            tty56           urandom         vofio
cpu_dma_latency mcelog          sda            tty            tty24           tty40           tty57           usbmon0         vga_arbiter
disk            md0            sda1           tty0            tty25           tty41           tty58           usbmon1         vhci
dm-0            mem            sda2           tty1            tty26           tty42           tty59           usbmon2         vhost-net
dm-1            memory_bandwidth sdb            tty10           tty27           tty43           tty6            vcs             vhost-vsock
dri            queue          sdb1           tty11           tty28           tty44           tty60           vcs1            zero
fb0            net            sdc            tty12           tty29           tty45           tty61           vcs2
fd             network_latency sdc1           tty13           tty3            tty46           tty62           vcs3
full           network_throughput sg0            tty14           tty30           tty47           tty63           vcs4
```



Ahora vamos a crear nuestro physical volumen a partir de md0

Hacemos

**sudo pvs**

Vemos que tenemos nuestro volumen group cl usando el physical volume /dev/sda. Pues vamos a crear como super usuario

**sudo pvcreate /dev/md0**

y comprobamos **sudo pvs** y ya tenemos nuestro physical volumen sin volumen group todavía.

```
[carmen@localhost ~]$ sudo pvs
[sudo] password for carmen:
PV          UG Fmt Attr PSize PFree
/dev/sda2   cl  lvm2 a-- <7,00g  0
[carmen@localhost ~]$ sudo pvcreate /dev/md0
Physical volume "/dev/md0" successfully created.
[carmen@localhost ~]$ sudo pvs
PV          UG Fmt Attr PSize PFree
/dev/md0     lvm2 --- <2,00g <2,00g
/dev/sda2    cl  lvm2 a-- <7,00g  0
[carmen@localhost ~]$ _
```

### Physical Volumen creado

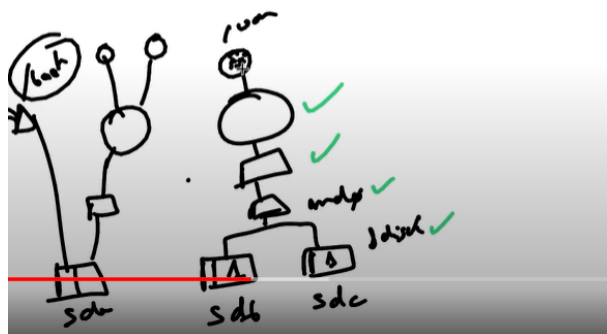
Ahora vamos a crear nuestro Volumen Group

**sudo vgcreate vg\_raid1 /dev/md0**

Y comprobamos con : **sudo vgs**

```
[carmen@localhost ~]$ sudo vgcreate vg_raid1 /dev/md0
Volume group "vg_raid1" successfully created
[carmen@localhost ~]$ sudo vgs
VG          #PV #LV #SN Attr   USize  UFree
cl           1  2   0 wz--n- <7,00g  0
vg_raid1     1  0   0 wz--n- <2,00g <2,00g
[carmen@localhost ~]$
```

Ahora tenemos cl con dos logical volumen y vg\_raid con ninguno, asique vamos a crear nuestro logical volumen.



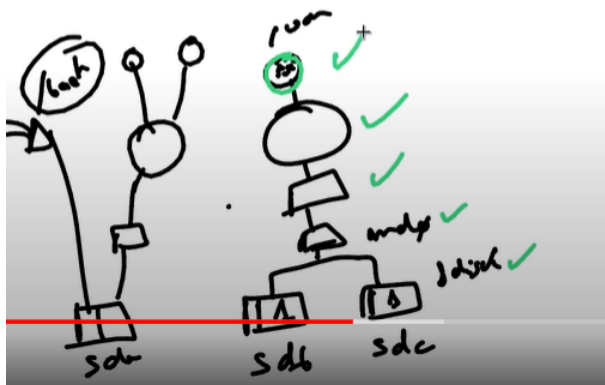
Vamos a crear nuestro volumen logico new\_var :

```
sudo lvcreate -n new_var -L 1.8G vg_raid1
```

le indicamos la capacidad 1.8, le dejamos un poco de espacio con respecto a la práctica anterior (2G) y le indicamos el volumen group donde tiene que crearlo debe ser nuestro RAID1.

Y comprobamos con **sudo lvs**

```
[carmen@localhost ~]$ sudo lvcreate -n new_var -L 1.8G vg_raid1
[sudo] password for carmen:
Rounding up size to full physical extent 1,80 GiB
Logical volume "new_var" created.
[carmen@localhost ~]$ sudo lvs
LV      VG      Attr      LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
root    cl      -wi-ao---- <6,20g
swap    cl      -wi-ao---- 820,00m
new_var vg_raid1 -wi-a----- 1,80g
[carmen@localhost ~]$ _
```



Ahora pues tenemos que hacer unos pasos similares a la práctica anterior, de traspaso de información, la parte del vm ya lo hemos hecho así que ahora tenemos que cifrar el volumen lógico, la herramienta que vamos a usar es cryptsetup, que tendremos que instalar previamente. En CentOS os vamos a usar dnf que es el reemplazo de yum. Necesitamos acceso a internet, con **ip addr** comprobamos que tenemos asignada una ip y comprobamos si tenemos acceso a fuera haciendo un ping a google. No podemos hacer esto ultimo, asi que vamos a levantar la interfaz enp0s3:

```
sudo ifup enp0s3
```

, comprobamos que tenemos internet y ahora ya si podemos acceder al exterior

```

Conexión activada con éxito (ruta activa D-Bus: /org/freedesktop/NetworkManager/ActiveConnection/1)
[carmen@localhost ~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:7a:0d:02 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
        valid_lft 86375sec preferred_lft 86375sec
    inet6 fe80::251f:8496:d001:1bd4/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[carmen@localhost ~]$ ping www.google.es
PING www.google.es (142.250.200.99) 56(84) bytes of data.
64 bytes from mad41s13-in-f3.1e100.net (142.250.200.99): icmp_seq=1 ttl=116 time=9.11 ms
64 bytes from mad41s13-in-f3.1e100.net (142.250.200.99): icmp_seq=2 ttl=116 time=9.97 ms
64 bytes from mad41s13-in-f3.1e100.net (142.250.200.99): icmp_seq=3 ttl=116 time=10.2 ms
64 bytes from mad41s13-in-f3.1e100.net (142.250.200.99): icmp_seq=4 ttl=116 time=11.1 ms
64 bytes from mad41s13-in-f3.1e100.net (142.250.200.99): icmp_seq=5 ttl=116 time=10.3 ms
64 bytes from mad41s13-in-f3.1e100.net (142.250.200.99): icmp_seq=6 ttl=116 time=11.3 ms
64 bytes from mad41s13-in-f3.1e100.net (142.250.200.99): icmp_seq=7 ttl=116 time=8.95 ms

```

Ya podemos acceder a los repositorios, vamos a hacer una búsqueda

**dnf search cryptsetup**

y ahí tenemos las herramientas, y pasamos a instalar cryptsetup

**sudo dnf install cryptsetup -y** (-y para que no pregunte)

Con este comando vamos a tener distintas acciones, open, format etc esta acción siempre va a venir precedida del prefijo luks, luks open, etc

Una vez instalado

```

Total 82 kB/s | 660 kB 00:00
advertencia:/var/cache/dnf/BaseOS-929b586ef1f72f69/packages/cryptsetup-2.3.3-4.el8.x86_64.rpm: Encabezado U3 RSA/SHA256 Signature, ID de clave 8483c65d: NOKEY
CentOS-8 - Base 191 kB/s | 1.6 kB 00:00
Importando llave GPG 0x8483C65D:
ID usuario: "CentOS (CentOS Official Signing Key) <security@centos.org>"
Huella : 99DB 70Fa E1D7 CE22 7FB6 4882 05B5 55B3 8483 C65D
Desde : /etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
La llave ha sido importada exitosamente
Ejecutando verificación de operación
Verificación de operación exitosa.
Ejecutando prueba de operaciones
Prueba de operación exitosa.
Ejecutando operación
Preparando : cryptsetup-libs-2.3.3-4.el8.x86_64 1/1
Actualizando : cryptsetup-libs-2.3.3-4.el8.x86_64 1/4
Ejecutando scriptlet: cryptsetup-libs-2.3.3-4.el8.x86_64 1/4
Actualizando : cryptsetup-2.3.3-4.el8.x86_64 2/4
Limpieza : cryptsetup-2.2.2-1.el8.x86_64 3/4
Limpieza : cryptsetup-libs-2.2.2-1.el8.x86_64 4/4
Ejecutando scriptlet: cryptsetup-libs-2.2.2-1.el8.x86_64 4/4
Verificando : cryptsetup-2.3.3-4.el8.x86_64 1/4
Verificando : cryptsetup-2.2.2-1.el8.x86_64 2/4
Verificando : cryptsetup-libs-2.3.3-4.el8.x86_64 3/4
Verificando : cryptsetup-libs-2.2.2-1.el8.x86_64 4/4
Actualizado:
cryptsetup-2.3.3-4.el8.x86_64 cryptsetup-libs-2.3.3-4.el8.x86_64
¡Listo!
[carmen@localhost ~]$

```

Vamos a darle formato al volumen lógico que hemos creado, vamos a cifrarlo, para ello:

**man cryptsetup para ver el manual**

Entonces:

**sudo cryptsetup luksFormat /dev/vg\_raid1/new\_var**

Nos pregunta si estamos seguros de sobrescribir de forma irrevocable, tenemos que escribir YES. Y ya nos pregunta por una contraseña, usamos como siempre practicasse y la verificamos y con esto ya tendríamos cifrado el volumen.

```
[carmen@localhost ~]$ sudo cryptsetup luksFormat /dev/vg_raid1/new_var
[sudo] password for carmen:

WARNING!
=====
Esto sobrescribirá los datos en /dev/vg_raid1/new_var de forma irrevocable.

Are you sure? (Type 'yes' in capital letters): YES
Introduzca la frase contraseña de /dev/vg_raid1/new_var:
Verifique la frase contraseña:
[carmen@localhost ~]$
```

Antes de cifrar la documentación de RedHat nos recomienda usar el comando shred, lo que hace es generar basura, para que? para dar un extra de seguridad y partir de una situación aleatoria. No lo hacemos ahora porque va a escribir los 2gb que tenemos y nos va a llevar tiempo, pero esto se haría como buena práctica llamar a shred antes de hacer luksFormat

Ya tenemos cifrado el disco, ahora vamos a acceder a él, para ello usamos de nuevo cryptsetup:

```
sudo cryptsetup luksOpen /dev/vg_raid1/new_var vg_raid1-new_var_crypt
```

Le pasamos la ruta y le damos un nombre para hacer referencia a ese volumen activado, (abierto puede ser un poco ambiguo). Vamos a usar la misma nomenclatura que usa mapper: **vg\_raid-new\_var\_crypt**

```
[carmen@localhost ~]$ sudo cryptsetup luksOpen /dev/vg_raid1/new_var vg_raid1-new_var_crypt
[sudo] password for carmen:
Introduzca la frase contraseña de /dev/vg_raid1/new_var:
[carmen@localhost ~]$ _
```

Para comprobar si ha funcionado o no usamos **ls /dev/mapper** y veremos como tenemos nuestro volumen cifrado y tendremos accesible nuestro volumen cifrado activado.

```
[carmen@localhost ~]$ ls /dev/mapper
cl-root cl-swap control vg_raid1-new_var vg_raid1-new_var_crypt
[carmen@localhost ~]$ _
```

ya tenemos nuestro diseño de la infraestructura y la hemos dejado lista. Los pasos siguientes son los mismo para montar /var que hicimos en la práctica 2. Luego si que habrá que hacer un paso final de configuración para indicarle al SO cuando arranque que tiene que activar este volumen cifrado, es decir que ejecuta directamente esta línea (**sudo cryptsetup luksOpen /dev/vg\_raid1/new\_var vg\_raid1-new\_var\_crypt**) y ahí un archivo especial para ello.

Pasamos a crear nuestro sistema de archivos.

```
sudo mkfs -t xfs /dev/mapper/vg_raid1-new_var_crypt
```

(\*\*Nota: Cuidado porque hay que decirle que lo cree en el volumen activado, si no no va a ser capaz de crear el sistema de archivo y recordamos también especificar el tipo)

Y ya pasamos al estado de mantenimiento con:

```
sudo systemctl isolate rescue.target
```

Por el bug, nos sacara del usuario, volvemos a entrar y ejecutar de nuevo

```
sudo systemctl isolate rescue.target
```

Si no nos deja cambiar el estado accedemos como root y volvemos a ejecutar el comando.

Ahora vamos a crear el punto de montaje /new\_var, montamos

```
[root@localhost ~]# mkdir /new_var
[root@localhost ~]# mount /dev/mapper/vg_raid1-new_var_crypt /new_var/
[ 5853.195432] XFS (dm-3): Mounting U5 Filesystem
[ 5853.209740] XFS (dm-3): Ending clean mount
[root@localhost ~]#
```

Copiamos /var en /new\_var y comprobamos con ls:

```
cp -a /var/. /new_var/
```

```
ls -laZ /new_var/
```

```
[root@localhost ~]# cp -a /var/. /new_var/
[root@localhost ~]# ls -laZ /new_var/
total 12
drwxr-xr-x. 21 root root system_u:object_r:var_t:s0      4096 nov  3 07:16 .
dr-xr-xr-x. 18 root root system_u:object_r:root_t:s0     239 nov  7 13:42 ..
drwxr-xr-x.  2 root root system_u:object_r:acct_data_t:s0 19 nov  3 07:13 account
drwxr-xr-x.  2 root root system_u:object_r:var_t:s0        6 may 10 2019 adm
drwxr-xr-x.  8 root root system_u:object_r:var_t:s0       91 nov  3 07:13 cache
drwxr-xr-x.  2 root root system_u:object_r:kdump_crash_t:s0 6 abr 24 2020 crash
drwxr-xr-x.  3 root root system_u:object_r:system_db_t:s0 18 nov  3 07:13 db
drwxr-xr-x.  3 root root system_u:object_r:var_t:s0       18 nov  3 07:13 empty
drwxr-xr-x.  2 root root system_u:object_r:public_content_t:s0 6 may 10 2019 ftp
drwxr-xr-x.  2 root root system_u:object_r:games_data_t:s0 6 may 10 2019 games
drwxr-xr-x.  2 root root system_u:object_r:var_t:s0        6 may 10 2019 gopher
drwxr-xr-x.  3 root root system_u:object_r:var_t:s0       18 nov  3 07:12 kerberos
drwxr-xr-x. 29 root root system_u:object_r:var_lib_t:s0    4096 nov  3 07:14 lib
drwxr-xr-x.  2 root root system_u:object_r:var_t:s0        6 may 10 2019 local
drwxrwxrwx.  1 root root system_u:object_r:var_lock_t:s0 11 nov  3 07:12 lock -> ../run/lock
drwxr-xr-x.  9 root root system_u:object_r:var_log_t:s0    233 nov  7 13:02 log
drwxrwxrwx.  1 root root system_u:object_r:mail_spool_t:s0 10 may 10 2019 mail -> spool/mail
drwxr-xr-x.  2 root root system_u:object_r:var_t:s0        6 may 10 2019 nis
drwxr-xr-x.  2 root root system_u:object_r:var_t:s0        6 may 10 2019 opt
drwxr-xr-x.  2 root root system_u:object_r:var_t:s0        6 may 10 2019 preserve
drwxrwxrwx.  1 root root system_u:object_r:var_run_t:s0    6 nov  3 07:12 run -> ../run
drwxr-xr-x.  7 root root system_u:object_r:var_spool_t:s0  66 nov  3 07:13 spool
drwxrwxrwt.  3 root root system_u:object_r:tmp_t:s0       33 nov  7 13:41 tmp
-rw-r--r--.  1 root root system_u:object_r:etc_runtime_t:s0 208 nov  3 07:12 .updated
drwxr-xr-x.  2 root root system_u:object_r:var_up_t:s0     6 may 10 2019 yp
[root@localhost ~]#
```

Y ahora indicamos la fstab donde queremos que monte:

**vi fstab**

y añadimos la línea

```
/dev/mapper/vg_raid1-new_var_crypt    /var    xfs    defaults    0 0
```

Por último hay que indicarle al SO cuando arranque que active ese volumen lógico. ¿Cómo se hace eso? Con el **archivo crypttab**

¿Cómo se especifica el dispositivo cifrado? con el uuid y esto se obtiene con blkid, filtramos la salida para quedarnos con los que están cifrados **blkid | grep crypt**

Cuidado porque queremos quedarnos con el que está cifrado con crypt luks, filtramos un poco mas la salida y la redirigimos a /etc/crypttab, que no existe

```
blkid | grep crypto > /etc/crypttab
```



y lo editamos, quitando el prefijo y añadimos y sufijo crypt, para indicar el volumen lógico activado, especificamos el userid del que no está cifrado y finalmente le indicamos que no utilice ningún archivo extra de forma que queda así:

```
vg_raid1-new_var_crypt UUID=2d398b88-abac-4078-83ba-2b1a57d503d3 none
```

Con esto ya estaría listo..

Vamos a hacer también a liberación de memoria

```
[root@localhost ~]# mv /var/ /var_old
[root@localhost ~]# mkdir /var
[root@localhost ~]# restorecon /var
[root@localhost ~]# _
```

Y **reboot**.

Al reiniciar nos pedirá la contraseña y con lsblk podemos comprobar que tenemos montado /var en un volumen cifrado que está en raid 1 con estos dos discos.

```
[carmen@localhost ~]# lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda                                  8:0      0   8G  0 disk
├─sda1                              8:1      0   1G  0 part  /boot
├─sda2                              8:2      0   7G  0 part
│   ├─cl-root                       253:0    0  6,2G  0 lvm    /
│   └─cl-swap                       253:1    0  820M  0 lvm    [SWAP]
sdb                                  8:16     0    2G  0 disk
├─sdb1                              8:17     0    2G  0 part
│   └─md0                           9:0      0    2G  0 raid1
│       └─vg_raid1-new_var          253:2    0  1,8G  0 lvm
│           └─vg_raid1-new_var_crypt 253:3    0  1,8G  0 crypt /var
sdc                                  8:32     0    2G  0 disk
├─sdc1                              8:33     0    2G  0 part
│   └─md0                           9:0      0    2G  0 raid1
│       └─vg_raid1-new_var          253:2    0  1,8G  0 lvm
│           └─vg_raid1-new_var_crypt 253:3    0  1,8G  0 crypt /var
sr0                                  11:0     1 1024M  0 rom
```

Ahora mismo swap no está cifrado entonces vamos, de forma que la información de var podría pasar en algún momento a RAM y de ram a swap, como configuración final podemos usar el comando swapoff, para desactivarlo y podemos decir con seguridad que esta configuración es correcta, en términos de seguridad.

**Fin**

## P2: Instalación de la pila LAMP en CentOS

Servidor Web: máquina para dar y recibir peticiones a través de internet.

Cuando se originó la web, se servían documentos html

Hypertext : texto que contiene hiperenlaces, hipervínculos

Markup: tiene etiquetas para definir características de ese texto.

Lenguaje

Los servimos con protocolos HTTP

URL (Universal /Uniform Resource Identifier) → Direcciones que no permiten identificar recursos web.

### LAMP

**L : linux**

**A : apache**

**M: MySQL**

**P php**

Vamos a ir instalando estos componentes en CentOS

### Apache

**dnf search apache**

Pasamos a instalarlo

**sudo dnf install httpd**

Conviene fijarnos en la version 2.4

Problema de siempre, could resolve host name bla bla → no tengo internet

**sudo ifup enp0s3 o sudo ip link set enp0s8 up**

Comprobamos que se ha instalado y lo habilitamos y lo activamos

**systemctl status httpd**

**sudo systemctl enable httpd**

**sudo systemctl start httpd**

```
[carmen@localhost ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: man:httpd.service(8)
[carmen@localhost ~]$ sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[carmen@localhost ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: man:httpd.service(8)
[carmen@localhost ~]$ sudo systemctl start httpd
[carmen@localhost ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2021-11-12 06:03:15 EST; 2s ago
     Docs: man:httpd.service(8)
  Main PID: 2068 (httpd)
   Status: "Started, listening on: port 80"
    Tasks: 213 (limit: 5019)
   Memory: 28.9M
```

Vamos a comprobar que efectivamente funciona. Comando **curl localhost**

y vemos que nuestro servidor httpd esta funcionando y nos devuelve esta pagina web

```
nd not your content. To prevent this page from ever being used, follow the instructions
<code>/etc/httpd/conf.d/welcome.conf</code>.</p>
<p>For systems using NGINX: You should now put your content in a location of your
edit the <code>root</code> configuration directive in the <strong>nginx</strong> config
<code>/etc/nginx/nginx.conf</code>.</p>
<p><a href="https://www.centos.org/"></a> </p>
</div>
<hr>
<div class="row">
  <div class="col">
    <h2 class="alert-heading">Important note!</h2>
    <p>The CentOS Project has nothing to do with this website or its content, it just
the software that makes the website run.</p>
    <p>If you have issues with the content of this site, contact the owner of the do
e CentOS project. Unless you intended to visit CentOS.org, the CentOS Project does not h
to do with this website, the content or the lack of it.</p>
    <p>For example, if this website is www.example.com, you would find the owner of
com domain at the following WHOIS server: <a href="http://www.internic.net/whois.html">h
ternic.net/whois.html</a></p>
  </div>
</div>
</main>
<footer class="container">
  <div>&#xA9; 2021 The CentOS Project | <a href="https://www.centos.org/legal/">Legal<
f="https://www.centos.org/legal/privacy/">Privacy</a></div>
</footer>
</body>
</html>
[carmen@localhost ~]$ _
```

## PHP

**dnf search php**

**sudo dnf install php**

Todo correcto. Ahora comprobamos que esta funcionando el intérprete: **php -a**

```
5
¡Listo!
[carmen@localhost ~]$ php -a
Interactive shell

php >
```

Por último, el ultimo modulo de LAMP,

## MySQL, mariadb

Mismo proceso:

**dnf search mariadb**

**sudo dnf install mariadb**

Estaremos instalando nuestro mysql database server

Comprobamos el estado del servicio y nos da un error: **systemctl status mariadb**

¿Por qué? Porque solo hemos instalado el cliente, entonces no tenemos ningún servicio activado. Si hacemos **dnf search mariadb** veremos que tenemos un paquete de server

Entonces tenemos que hacer también:

```
sudo dnf install mariadb-server
```

Comprobamos el estado del servicio, y vemos que esta desactivado y deshabilitado:

```
sudo systemctl enable mariadb
```

```
sudo systemctl start mariadb
```

```
[carmen@localhost ~]$ systemctl status mariadb
● mariadb.service - MariaDB 10.3 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
[carmen@localhost ~]$ sudo systemctl enable mariadb
Created symlink /etc/systemd/system/mysql.service → /usr/lib/systemd/system/mariadb.service.
Created symlink /etc/systemd/system/mysqld.service → /usr/lib/systemd/system/mariadb.service.
Created symlink /etc/systemd/system/multi-user.target.wants/mariadb.service → /usr/lib/systemd/system/mariadb.service.
[carmen@localhost ~]$ sudo systemctl start mariadb
[carmen@localhost ~]$
```

```
[carmen@localhost ~]$ sudo systemctl status mariadb
● mariadb.service - MariaDB 10.3 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2021-11-12 06:15:34 EST; 32s ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
   Process: 5851 ExecStartPost=/usr/libexec/mysql-check-upgrade (code=exited, status=0/SUCCESS)
   Process: 5716 ExecStartPre=/usr/libexec/mysql-prepare-db-dir mariadb.service (code=exited, status=0/SUCCESS)
   Process: 5692 ExecStartPre=/usr/libexec/mysql-check-socket (code=exited, status=0/SUCCESS)
   Main PID: 5819 (mysqld)
    Status: "Taking your SQL requests now..."
```

Y ahora viene un paso muy importante.

Si intentamos conectarnos a mysql,

**mysql**

no nos dejará, probamos con root

**mysql -u root**

```
[carmen@localhost ~]$ mysql
ERROR 1045 (28000): Access denied for user 'carmen'@'localhost' (using password: NO)
[carmen@localhost ~]$ mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.3.28-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]>
```

y de repente tengo acceso como usuario root, sin introducir contraseña. Cuidado porque si no leemos la documentación podemos cometer fallos. Lo que indica la documentación es que después de instalar tenemos que ejecutar el mysql secure installation qué es un script que lo que hace es ajustar algunos parámetros que vienen por defecto de una manera insegura. **ma n mysql\_secure\_installation**

lo ejecutamos: **mysql\_secure\_installation**

Le damos a intro, Y para ponerle una contraseña a root, que actualmente no tiene ( desde fuera alguien podría conectarse ahora mismo), pondremos **practicasis** como contraseña.

**Tenemos un usuario anónimo**, lo quitamos

**Disallow root login remotely?** Y, deshabilitamos el acceso a root remoto

**Remove test database and access to it?** Y, eliminamos la bd de prueba

**Reload privilege tables now?** Y, recargamos la tabla de privilegios (todo Y realmente)

Y con esto tenemos nuestra bd de mariadb configurada

Y comprobamos.

**mysql** no nos deja

**mysql -u root -p** nos deja entrar como root y nos pide la contraseña

```
All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
[carmen@localhost ~]$ mysql
ERROR 1045 (28000): Access denied for user 'carmen'@'localhost' (using password: NO)
[carmen@localhost ~]$ mysql -u root
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
[carmen@localhost ~]$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 21
Server version: 10.3.28-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> _
```

Ahora vamos a ver si somos capaces de conectarnos con apache.

Nos vamos a una terminal desde fuera:

**curl 192.168.56.110**

no nos dejará. Comprobamos que tenemos conexión haciendo un ping, que si nos dejara

```
C:\Users\carme>curl 192.168.56.110
curl: (7) Failed to connect to 192.168.56.110 port 80: Timed out

C:\Users\carme>ping 192.168.56.110

Haciendo ping a 192.168.56.110 con 32 bytes de datos:
Respuesta desde 192.168.56.110: bytes=32 tiempo<1m TTL=64
Respuesta desde 192.168.56.110: bytes=32 tiempo<1m TTL=64
Respuesta desde 192.168.56.110: bytes=32 tiempo<1m TTL=64
```

Que puede estar causando ese problema? El cortafuegos que está impidiendo que tengamos acceso al puerto. Lo añadimos de forma permanente y ya hacemos la recarga y probamos que tenemos conexión.

**sudo firewall-cmd --add-port=88/tcp**

**sudo firewall-cmd --add-port=88/tcp --permanent**

**sudo firewall-cmd --reload**

```
[carmen@localhost ~]$ sudo firewall-cmd --add-port=88/tcp
[sudo] password for carmen:
success
[carmen@localhost ~]$ sudo firewall-cmd --add-port=88/tcp --permanent
success
[carmen@localhost ~]$ sudo firewall-cmd --reload
success
[carmen@localhost ~]$
```

No va desde fuera no se porq

Ahora nos queda solo ver que podemos integrar un script de php q se conecta a una base de datos y el resultado lo muestra a través de un archivo html

Para ello vamos a buscar php mysql connect en google y el primer enlace que aparece a la documentación y vemos el ejemplo que hay abajo

```
<?php
$enlace = mysqli_connect('localhost', 'mi_usuario', 'mi_contraseña',
'mi_bd');

if (!$enlace) {
    die('Error de Conexión (' . mysqli_connect_errno() . ') '
        . mysqli_connect_error());
}

echo 'Éxito... ' . mysqli_get_host_info($enlace) . "\n";

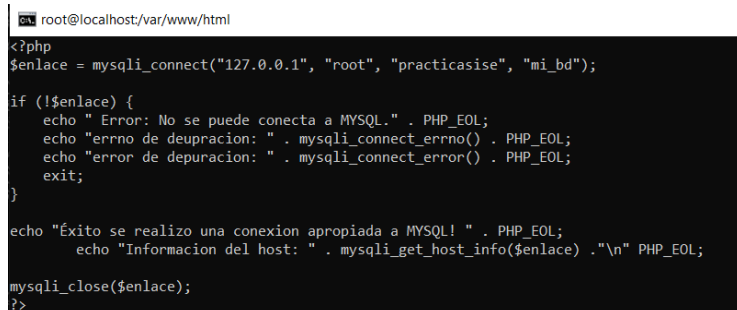
mysqli_close($enlace);
?>
```

Vamos a incluir esto en la máquina.

Para esto he accedido desde la terminal de fuera con ssh ( he tenido que cambiar el archivo sshd\_config en centOs para que me dejara acceder con root desde fuera y reiniciar el servicio ssh)

Un vez dentro desde la terminal, hacemos

```
cd /var/www/html/
sudo vi index.php
```



```
root@localhost:var/www/html
<?php
$enlace = mysqli_connect("127.0.0.1", "root", "practicasis", "mi_bd");

if (!$enlace) {
    echo " Error: No se puede conecta a MYSQL." . PHP_EOL;
    echo "errno de deupracion: " . mysqli_connect_errno() . PHP_EOL;
    echo "error de depuracion: " . mysqli_connect_error() . PHP_EOL;
    exit;
}

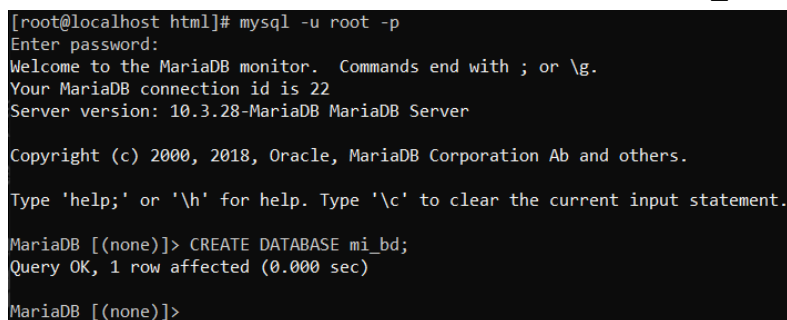
echo "Éxito se realizo una conexion apropiada a MYSQL! " . PHP_EOL;
    echo "Informacion del host: " . mysqli_get_host_info($enlace) . "\n" PHP_EOL;

mysqli_close($enlace);
?>
```

La base de datos no la tenemos aún, vamos a crear.

Nos conectamos como usuario root: **mysql -u root -p**

Y creamos la base de datos: **CREATE DATABASE mi\_bd;**



```
[root@localhost html]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 22
Server version: 10.3.28-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> CREATE DATABASE mi_bd;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]>
```

Ahora vamos a comprobar que funciona correctamente. Nos vamos al navegador y tecleamos nuestra ip y debería ir pero no va u.u

Si pedimos ip/index.php nos sacaría el script en modo texto. **Qué ocurre?** Que apache no está interpretando php, solo devolviendo el documento. Para ello vamos a cambiar la configuración de php para indicarle que interprete este archivo.

Nos vamos centOs y editamos el archivo de configuración de httpd :

**sudo vi /etc/httpd/conf/httpd.conf**

**/Directo** : para buscar

**y le indicamos en el directorio index sirva cualquier archivo con extensión .php**

```
#
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#
<IfModule dir_module>
    DirectoryIndex index.html *.php
</IfModule>
#
```

Y reiniciamos el servicio httpd

Y al recargar el navegador nos da un error 500, le damos a inspeccionar y nos dice que el servidor ha tenido un error.

Vamos a ver qué ha podido pasar. Vemos si podemos ejecutar nuestro archivo desde el servidor:

- **cd /var/www/html/**
- **php index.php**

Y nos dice que hay una función que no conoce, porque no hemos instalado la biblioteca que nos comunica php con mysql para ello hacemos:

**sudo dnf search php | grep mysql**

**sudo dnf install php-mysqldb**

Y comprobamos que desde el servidor si podemos ejecutar el fichero index.php

```
[carmen@localhost ~]$ cd /var/www/html/
[carmen@localhost html]$ php index.php
Éxito se realizó una conexión apropiada a MySQL!
Información del host: 127.0.0.1 via TCP/IP
[carmen@localhost html]$
```

En el navegador tenemos un problema aun de permisos. Echamos un vistazo a :

**sudo less /var/log/audit/audit.log**

**SELinux tiene una lista de booleanos que podemos consultar con**  
**getsebool -a | grep httpd**

y hay uno en concreto: httpd\_can\_network\_connect\_db que esta a off

Hay que ponerlo a on:

**sudo setsebool -P httpd\_can\_network\_connect\_db=on**

**Y ya debería funcionar.**