

ISEPractica1.pdf



S_GRND



Ingeniería de Servidores



3º Grado en Ingeniería Informática



Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación
Universidad de Granada

Aprende Inglés
Con nuestros cursos **GRATUITOS**
para desempleados

**Pincha aquí e
inscríbete ya**

Fórmate con nuestros cursos de Inglés para las titulaciones
A1, A2, B1 y B2 100€ subvencionados para desempleados



GOBIERNO
DE ESPAÑA

MINISTERIO
DE TRABAJO
Y ECONOMÍA SOCIAL

SERVICIO PÚBLICO
DE EMPLEO ESTATAL
SEPE



Aprende Inglés

Con nuestros cursos **GRATUITOS**
para desempleados



Fórmate con nuestros cursos de Inglés para las titulaciones
A1, A2, B1 y B2 100€ subvencionados para desempleados

Pincha aquí e
inscríbete ya

958 047 283
621 21 76 50



GOBIERNO
DE ESPAÑA

MINISTERIO
DE TRABAJO
Y ECONOMÍA SOCIAL

SERVICIO PÚBLICO
DE EMPLEO ESTATAL
SEPE



ANDALUCÍA EMPRENDE,
FUNDACIÓN PÚBLICA ANDALUZA
Consejería de Empleo,
Formación y Trabajo Autónomo



Mac

10 %
dto | Estudiantes
Profesores



iPad

6 %
dto | Estudiantes
Profesores



Rossellimac®

PRACTICA 1

VIRTUALIZACION SOFTWARE – HIPERVISOR

- Plataforma
- Seguridad/encapsulado
- Optimización de recursos
- Económica

CLOUD COMPUTING

Contenedor: no virtualiza. Tiene el mismo hw y so que el anfitrión. Mejora el rendimiento de la aplicación que corramos.

Namespaces

RAID:

- 0: junta el espacio de varios discos. Así, no se tiene dividido el espacio. Más caro. Almacena los ficheros de forma distribuida entre los discos. Si se rompe un disco puedes perder mucha información.
- 1: escribe la misma información entre los distintos discos -> redundancia. Si se rompe un disco el sistema sigue funcionando. Lecturas pueden hacerse en paralelo. Este es lógico con varios buses
- 5: con bit de paridad para evitar perdida de información cuando uno de los discos cae

En LVM tendremos espacios de almacenamiento llamados phisical volumen -> los agruparemos en volumen group. Iremos dando mas almacenamiento a los distintos archivos/directorios/particiones (swap, home, root...).

LECCION 1

Se nos contextualiza en una empresa proveedora de servicios y nos solicitan crear un VPS para la implantación de un comercio electrónico que utilizará un CMS (sistema de gestión de contenidos). No tenemos muchos detalles, pero nos recomiendan una configuración: tendrá RAID1 (nos duplicará la información) y estará gestionado con LVM. También se recomienda que cifremos la información para cumplir con la legislación vigente. Se nos recomienda 3 volúmenes lógicos (uno para hogar, otro para la raíz y otro para el swap) y una partición separada para el arranque

Creamos la maquina virtual en VBox.

Si nos pide RAID1, ¿Cuál es el numero mínimo de discos que necesitaremos para tener RAID1? 2. Cuando escribamos en un disco, automáticamente se escribirán en los demás discos del RAID1. Aumentando el nro de discos, aumentamos el coste, el espacio y la alimentación (energía) pero esta disminuyendo la posibilidad de fallos. No hay un máximo para el número de discos, pero si un mínimo: 2. Por tanto, vamos a crear un nuevo disco virtual en la máquina. (Configuración -> Almacenamiento ->SATA -> icono de disco duro)

Arrancamos la máquina. Continuamos pasos obvios (sin actualizar, español, etc). Hasta llegar a “Guided Storage Configuration”. Aquí podremos especificar todos los requisitos pedidos en el guión.

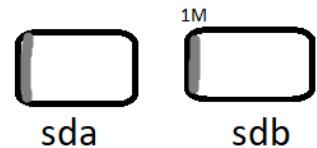
1. Vamos a definir un almacenamiento a medida: Seleccionamos “Custom Storage Layout”

2. Aquí vemos los dos discos que tenemos creados, de igual tamaño. Tenemos que ser conscientes de las especificaciones que nos han pedido.

Una de las cosas que nos piden es hacer una partición para el arranque. Esto significa que tendremos nuestros discos físicos sda y sdb, y sobre ellos tendremos una capa lógica de división (gracias a las particiones podremos dividir a nivel

lógico los discos en distintos compartimentos, para distribuir mejor la información.) Tenemos que asegurarnos de que dejamos espacio para cargar el sector de arranque en ambos discos porque si sda fallase por algún motivo, sdb debería ser capaz de arrancar. Por tanto, en ambos vamos a querer instalar el GRUB (lo configuraremos para que se haga de forma automática). Vamos a dejar 1MB para él de forma manual.

DISPOSITIVOS DISPONIBLES		
DISPOSITIVO	TIPO	TAMAÑO
[VBOX_HARDDISK_VB4d548cd0-8b621714]	disco local	10.000G ►]
unused		
[VBOX_HARDDISK_VBf103ae77-fa67c24f]	disco local	10.000G ►]
unused		
[Create software RAID (md) ►]		
[Crear grupo de volúmenes (LVM) ►]		
DISPOSITIVOS UTILIZADOS		
No used devices		



Entonces vamos a crear luego una partición para nuestro arranque -> vamos a mapear /boot. Vamos a pensar antes qué espacio tenemos que dejarle -> Nosotros le ajustamos 400M (porque así podemos tener dos versiones del kernel)

Vamos a crear por tanto esa partición en la máquina. Pulsamos en uno de los discos -> Add GPT partition. Le ponemos un tamaño de 400M y lo dejamos sin formato. Si nos fijamos el instalador ha sido muy astuto y nos ha reservado 1M para el GRUB.

```

DISPOSITIVO          TIPO      TAMAÑO
[ VBOX_HARDDISK_VB4d548cd0-8b621714 disco local  10.000G ▶ ]
partition 2 new, unused        400.000M ▶
espacio disponible           9.606G

[ VBOX_HARDDISK_VBf103ae77-fa67c24f disco local  10.000G ▶ ]
unused

[ Create software RAID (md)      ▶ ]
[ Crear grupo de volúmenes (LVM) ▶ ]

DISPOSITIVOS UTILIZADOS

DISPOSITIVO          TIPO      TAMAÑO
[ VBOX_HARDDISK_VB4d548cd0-8b621714 disco local  10.000G ▶ ]
partition 1 new, bios_grub    1.000M ▶

```

Para el segundo disco vamos a definir esa partición. Podemos crearla a mano, de tamaño 1M, para el GRUB. O directamente podemos indicarle que utilizaremos ese dispositivo para el arranque -> "Add As Another Boot Device". Pulsamos y vemos como automáticamente nos ha creado esa partición con ese mega extra.

```

DISPOSITIVO          TIPO      TAMAÑO
[ VBOX_HARDDISK_VB4d548cd0-8b621714 disco local  10.000G ▶ ]
partition 2 new, unused        400.000M ▶
espacio disponible           9.606G

[ VBOX_HARDDISK_VBf103ae77-fa67c24f disco local  10.000G ▶ ]
espacio disponible           9.997G

[ Create software RAID (md)      ▶ ]
[ Crear grupo de volúmenes (LVM) ▶ ]

DISPOSITIVOS UTILIZADOS

DISPOSITIVO          TIPO      TAMAÑO
[ VBOX_HARDDISK_VB4d548cd0-8b621714 disco local  10.000G ▶ ]
partition 1 new, bios_grub    1.000M ▶
[ VBOX_HARDDISK_VBf103ae77-fa67c24f disco local  10.000G ▶ ]
partition 1 new, bios_grub    1.000M ▶

```

Ahora creamos otra vez la partición para boot en el segundo disco -> 400M y sin formato. (Sin formato porque queremos nuestra configuración RAID -> lo queremos tanto para el volumen LVM como para el arranque).

```

DISPOSITIVO          TIPO      TAMAÑO
[ VBOX_HARDDISK_VB4d548cd0-8b621714 disco local  10.000G ▶ ]
partition 2 new, unused        400.000M ▶
espacio disponible           9.606G

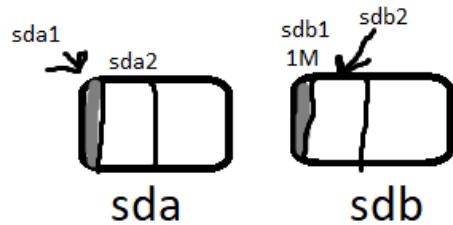
[ VBOX_HARDDISK_VBf103ae77-fa67c24f disco local  10.000G ▶ ]
partition 2 new, unused        400.000M ▶
espacio disponible           9.606G

[ Create software RAID (md)      ▶ ]
[ Crear grupo de volúmenes (LVM) ▶ ]

DISPOSITIVOS UTILIZADOS

DISPOSITIVO          TIPO      TAMAÑO
[ VBOX_HARDDISK_VB4d548cd0-8b621714 disco local  10.000G ▶ ]
partition 1 new, bios_grub    1.000M ▶
[ VBOX_HARDDISK_VBf103ae77-fa67c24f disco local  10.000G ▶ ]
partition 1 new, bios_grub    1.000M ▶

```



3. Lo que vamos a hacer ahora es, a raíz de la partición de sda2 y a partir de sdb2, crearemos una primera abstracción para nuestro RAID y se denominará md0. Con respecto a los RAID, tenemos la posibilidad del RAID Hardware o Software -> Nosotros vamos a hacer Software de tipo 1 (RAID1)

Vamos a “Create software RAID (md)” -> dejamos nombre md0 -> especificamos nivel de RAID (1). Seleccionamos la partición que queremos -> Escogemos las particiones de 400M.

Ahora vemos como tenemos nuestro nuevo dispositivo md0 -> tamaño de 398 (le hemos dado 400 pero con los metadatos que necesita el software RAID se queda en 398)

DISPOSITIVOS DISPONIBLES		
DISPOSITIVO	TIPO	TAMAÑO
[md0 (new) unused	software RAID 1	398.000M ►]

4. Ahora vamos a pasar a crear nuestro RAID1 para la partición que va a contener el resto de los datos (la tercera partición de nuestros dos discos). Primero creamos la partición con el resto de espacio libre en cada uno de los discos. No especificamos tamaño. Lo dejamos sin formatear (en el momento en el que escogamos un sistema de archivos, ya no podremos definir un RAID1). Al no especificar el numero estamos escogiendo el tamaño máximo.

DISPOSITIVO	TIPO	TAMAÑO
[md0 (new) unused	software RAID 1	398.000M ►]
[VBOX_HARDDISK_VB4d548cd0-8b621714 partition 3 new, unused	disco local	10.000G ►] 9.606G ►]
[VBOX_HARDDISK_VBf103ae77-fa67c24f partition 3 new, unused	disco local	10.000G ►] 9.606G ►]

Ya tenemos la partición 3 creada.

5. Ahora vamos a pasar a crear nuestra abstracción lógica md1 de manera análoga a antes. Creamos nuestro software RAID con nombre md1. Seleccionamos las dos particiones gemelas de 9,6G.



Mac
10 % dto | Estudiantes Profesores



iPad
6 % dto | Estudiantes Profesores



Rossellimac®

```

DISPOSITIVOS DISPONIBLES
[ DISPOSITIVO           TIPO          TAMAÑO
[ md1 (new)           software RAID 1   9.597G ▶ ]
unused

[ md0 (new)           software RAID 1   398.000M ▶ ]
unused

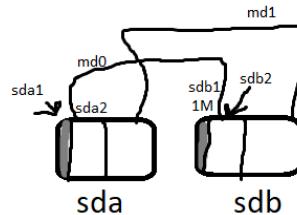
[ Create software RAID (md) ▶ ]
[ Crear grupo de volúmenes (LVM) ▶ ]

DISPOSITIVOS UTILIZADOS
[ DISPOSITIVO           TIPO          TAMAÑO
[ VBOX_HARDDISK_VB4d548cd0-8b621714 disco local   10.000G ▶ ]
partition 1 new, bios_grub           1.000M ▶
partition 2 new, component of software RAID 1 md0   400.000M ▶
partition 3 new, component of software RAID 1 md1   9.606G ▶

[ VBOX_HARDDISK_VBf103ae77-fa67c24f  disco local   10.000G ▶ ]
partition 1 new, bios_grub           1.000M ▶
partition 2 new, component of software RAID 1 md0   400.000M ▶
partition 3 new, component of software RAID 1 md1   9.606G ▶

```

Si nos fijamos ya no nos sale en dispositivos disponibles sda y sdb (ya no tenemos acceso a ellos en esa sección), si no que tenemos directamente las abstracciones de los dispositivos RAID1 (md0 y md1).



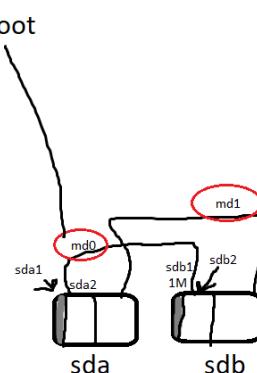
- Vamos a pasar a asignarle el punto de montaje y el formato a la partición del arranque (md0). Pulsamos en “md0” y seleccionamos “Format”. -> vamos a montar /boot -> dejamos ext4 (los demás no tienen sentido porque son sistemas de archivos pensados para archivos mas grandes).

```

RESUMEN DEL SISTEMA DE ARCHIVOS

PUNTO DE MONTAJE    TAMAÑO    TIPO    TIPO DE DISPOSITIVO
[ /boot              398.000M new ext4  new software RAID 1 ▶ ]

```



7. Vamos a pasar ahora a crear la configuración que se nos exige en el enunciado en cuanto a los volúmenes lógicos (LVM).

Los volúmenes lógicos son una abstracción. Tenemos un dispositivo físico, ese dispositivo físico se abstrae y se crea un PV (una abstracción lógica). Esta abstracción lógica (los volúmenes físicos, PV) se va a agrupar en un grupo de volúmenes -> VG. Ese grupo de volúmenes que agrupa PV, se puede desagregar en LV (logical volume). Tantos como queramos: en nuestro caso tendremos uno para la raíz (/) otro para hogar (/home) y otro para swap. (3 volúmenes lógicos). Todo esto se conoce como LVM (logical volume manager)

Pero, ¿por qué partimos de desagregados, agrupamos y volvemos a desagregar? ¿Qué ventajas tiene esto?

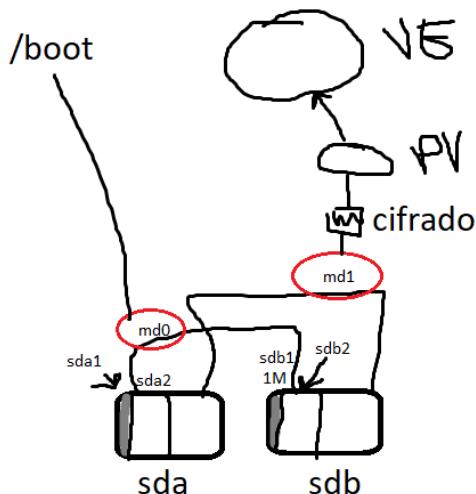
1. Nos da flexibilidad. Tratando con los dispositivos físicos nos costaría mucho más, ya que a nivel lógico podemos borrar, eliminar, redimensionar... (dentro de las posibilidades de los sistemas de archivos). Además, podemos añadir o quitar PV al volumen group, con lo cual si nos quedamos sin espacio, podemos comprar más discos, creamos su PV correspondiente, añadimos ese PV al grupo de volúmenes y podemos incrementar los volúmenes lógicos sin redefinir el esquema que tenemos de particionado. Si se nos rompe o nos sobra, podemos eliminarlo y todo lo hacemos de forma transparente.
2. Snapshots: Instantáneas de los volúmenes lógicos que nos ayudarán con las copias de seguridad
3. Etc

Dicho esto, vamos a proceder a configurar el sistema con lo que se nos pide. Vamos a crear nuestro grupo de volúmenes partiendo de un dispositivo físico. -> "Crear grupo de volúmenes". Dejamos el nombre en vg0. Escogemos el dispositivo "md1".

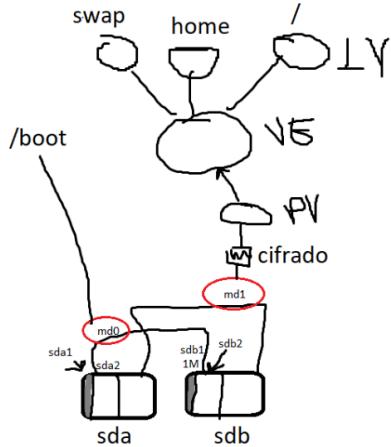
Queremos cifrar la información. /boot no va a ir cifrado pero el grupo de volúmenes y lo que nos está almacenando md1 si queremos que vaya cifrado. Marcamos "Crear volumen cifrado" y ponemos contraseña.

DISPOSITIVO	TIPO	TAMAÑO
[vg0 (new, encrypted) unused	LVM volume group	9.578G ▶]

Ya tenemos nuestro grupo de volúmenes cifrado. Esto lo que ha hecho ha sido cifrar nuestro dispositivo md1, crear la abstracción de PV, crear el grupo de volúmenes vg0 y añadirle a este grupo de volúmenes nuestro PV



8. Ahora vamos a crear los 3 VL correspondientes para swap, home y raíz (/). Queremos que quede así:



Pinchamos en vg0 y luego en “Create Logical Volume”.

- Swap: nombre “swap”, ¿qué tamaño? 1G. Vamos a indicarle en el formato que va a ser el espacio de intercambio. Lo creamos.

PUNTO DE MONTAJE	TAMAÑO	TIPO	TIPO DE DISPOSITIVO
/boot	398.000M	new ext4	new software RAID 1 ►]
SWAP	1.000G	new swap	new LVM logical volume ►]



- Home: nombre “hogar”. Tendremos en cuenta que la gran mayoría de archivos, bases de datos, etc van a ir a /Var. ¿Tendría sentido crear un LV para /Var? Si. Pero en este caso vamos a dejar /Var en la raíz (/). Por tanto raíz va a ser el LV con mas tamaño. A home por tanto le daremos 500M. Le dejamos el formato en ext4 porque no parece que vaya a tener archivos excesivamente grandes. Le asignamos el punto de montaje en /home. Lo creamos

PUNTO DE MONTAJE	TAMAÑO	TIPO	TIPO DE DISPOSITIVO
/boot	398.000M	new ext4	new software RAID 1 ►]
/home	500.000M	new ext4	new LVM logical volume ►]
SWAP	1.000G	new swap	new LVM logical volume ►]



- Raíz: nombre “raíz”. No le especificamos tamaño para dejarle los 8GB restantes. En formato igual tendría sentido los sistemas con archivos mas grandes pero vamos a dejarlo en ext4. El punto de montaje lo dejamos en /. Lo creamos.

PUNTO DE MONTAJE	TAMAÑO	TIPO	TIPO DE DISPOSITIVO
/	8.089G	new ext4	new LVM logical volume ►]
/boot	398.000M	new ext4	new software RAID 1 ►]
/home	500.000M	new ext4	new LVM logical volume ►]
SWAP	1.000G	new swap	new LVM logical volume ►]



Ya tenemos creado el esquema que nos pide.

¿Boot va cifrado? No, porque la actual implementación de GRUB2 no lo permite. De un modo avanzado si podríamos. Pero no es necesario cifrarlo porque en root tenemos la imagen del kernel

y no pasaría nada que tuviéramos una brecha de seguridad y alguien tuviese acceso (no tendremos información privada). En cambio en raíz o en /var si vamos a tener información protegida.

Ya podemos proceder a “Hecho”. Nos advierte de las modificaciones que hemos hecho. Continuamos. Añadimos nuestro usuario.

No instalamos el servidor SSH. Tampoco los demás servicios. Ahora esperamos a que termine la configuración y reiniciamos la máquina para comprobar que lo hemos hecho correctamente.

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
loop0	7:0	0	29.9M	1	loop	/snap/snapd/8542
loop1	7:1	0	55M	1	loop	/snap/core18/1880
loop2	7:2	0	71.3M	1	loop	/snap/1xd/16099
sda	8:0	0	10G	0	disk	
└─sda1	8:1	0	1M	0	part	
└─sda2	8:2	0	400M	0	part	
└─md0	9:0	0	399M	0	raid1	/boot
└─sda3	8:3	0	9.6G	0	part	
└─md1	9:1	0	9.6G	0	raid1	
└─dm_crypt-0	253:0	0	9.6G	0	crypt	
└─vg0-swap	253:1	0	1G	0	lvm	[SWAP]
└─vg0-hogar	253:2	0	500M	0	lvm	/home
└─vg0-raiz	253:3	0	8.1G	0	lvm	/
sdb	8:16	0	10G	0	disk	
└─sdb1	8:17	0	1M	0	part	
└─sdb2	8:18	0	400M	0	part	
└─md0	9:0	0	399M	0	raid1	/boot
└─sdb3	8:19	0	9.6G	0	part	
└─md1	9:1	0	9.6G	0	raid1	
└─dm_crypt-0	253:0	0	9.6G	0	crypt	
└─vg0-swap	253:1	0	1G	0	lvm	[SWAP]
└─vg0-hogar	253:2	0	500M	0	lvm	/home
└─vg0-raiz	253:3	0	8.1G	0	lvm	/
sr0	11:0	1	1024M	0	rom	



Mac
10 % dto | Estudiantes Profesores



iPad
6 % dto | Estudiantes Profesores



Rossellimac[®]

LECCION 2

Man lsblk-> para consultar el manual (dudas, repasar, aprender parámetros nuevos)

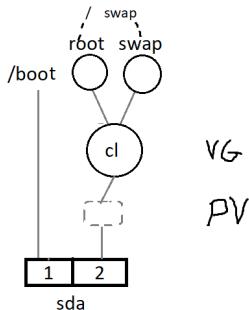
Lsblk -> para ver el esquema de discos

```
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]
sr0     11:0   1 58,3M 0 rom
```

Vemos que al principio tenemos el disco sda y centOs nos ha creado automáticamente una partición de 1GB (sda1) donde tiene /boot asignado como punto de montaje y luego tenemos otra partición de los 7GB restantes (sda2) donde tenemos gestionado por VLM un grupo de volúmenes denominados cl:

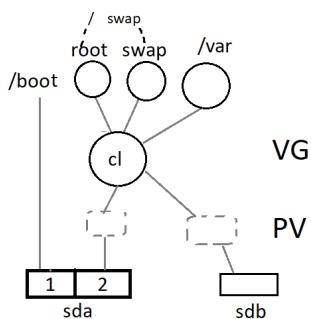
Dos volumentes lógicos: cl-root y cl-swap

Gráficamente:



Esta es la configuración inicial, la que tenemos por defecto al instalar la máquina virtual.

Tenemos que pasar al siguiente esquema:

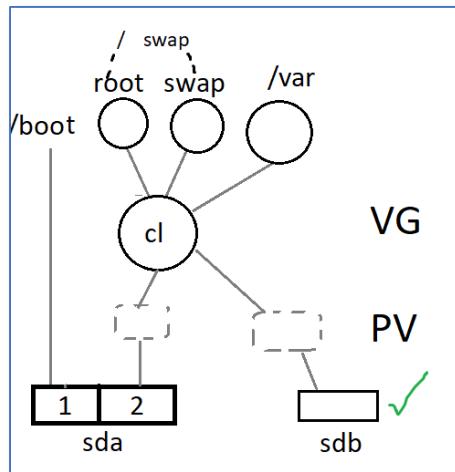


Vamos a tener sda (lo mismo que antes), pero ahora lo nuevo es añadir nuestro sdb (comentaremos si lo particionamos o no), crearemos su phisical volumen correspondiente, se lo añadiremos a cl y a partir de cl crearemos un nuevo volumen lógico para asignarle /var.

1. Creamos un nuevo disco desde la máquina virtual (Configuración, Almacenamiento, Sata, Icono de disco duro de la derecha del todo, Crear, Continuar todos los pasos)
- Vamos a comprobar si no ha habido un error:

```
[root@calhost ~]# 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
  └─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
sr0     11:0    1 58,3M 0 rom
```

Como vemos, está sdb -> se ha creado bien



2. Podemos continuar creando nuestro Phisical Volumen utilizando todo el disco (sdb) o podemos crear una partición. Las dos formas son posibles -> puede ser recomendable crear una partición para tener un espacio para metadatos o si en algún momento determinado se va a hacer una instalación.

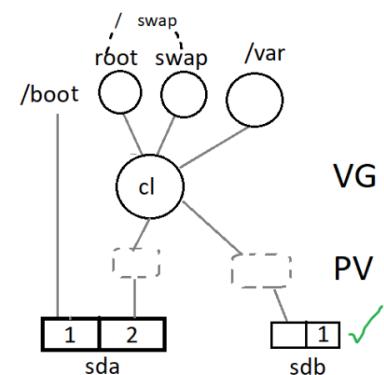
Vamos a utilizar fdisk

Ponemos “sudo fdisk /dev/sdb” -> como super usuario vamos a darle formato a /dev/sdb nos autenticamos y nos aparece un menú

- m -> ver todas las opciones
- p -> ver particiones
- n -> crear una partición

Ponemos “n”, en este caso la haremos primaria -> ponemos “p”. Le asignamos el numero “1” de partición -> Ponemos “1” -> Ponemos valor predeterminado/por defecto -> No ponemos nada -> Igual con el ultimo sector.

Cuando tecleamos ahora p para ver la tabla de particiones vemos que tenemos nuestra partición sdb1 y con w pasamos a escribirla. Ahora cuando ponemos “lsblk” tenemos nuestra partición sdb1.



```

Orden (m para obtener ayuda): p
Disco /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 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: 0xc37ccf05

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): 1
Primer sector (2048-16777215, valor predeterminado 2048):
Último sector, +sectores o +tamaño{K,M,G,T,P} (2048-16777215, valor predeterminado 16777215):

Crea una nueva partición 1 de tipo 'Linux' y de tamaño 8 GiB.

```

```

Orden (m para obtener ayuda): w
Se ha modificado la tabla de particiones.
Llamando a ioctl() para volver a leer la tabla de particiones.
[ 818.477239] sdb: sdb1
Se están sincronizando los discos.

```

```

[ 818.477239] 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   8G  0 disk
└─sdb1   8:17   0   8G  0 part
sr0     11:0    1 58,3M 0 rom

```

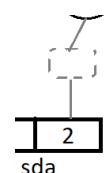
3. Vamos a pasar a crear nuestro phisical volumen. Dos posibilidades

- Ejecutar el command line interface de vlm (“sudo vlm”) en el que si presionamos tab dos veces vemos todos los comandos disponibles de vlm
- Invocar estos comandos desde el terminal sin tener esa miniconsola.

Para obtener información sobre los phisical volumen tendremos que poner “pvdisplay”

Recordemos que siempre tendremos que ejecutar como superusuario, es decir “sudo pvdisplay”

Aquí tenemos la información del phisical volumen y el nombre que recibe. Ya sabemos el nombre del rectángulo de líneas discontinuas que teníamos en el dibujo: /dev/sda2 que pertenece al grupo de volúmenes (VG) al que pertenece.



```

[ 818.477239] localhost ~]$ sudo pvdisplay
--- Physical volume ---
PU Name          /dev/sda2
VG Name          cl
PV 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
PV UUID          Xuc1KE-0dkH-7Uaq-Ake0-bksb-qGaa-ghqksp

```

A modo resumido podemos poner “sudo pvs”

```

[ 818.477239] localhost ~]$ sudo pvs
[sudo] password for :
PV          VG Fmt Attr PSize  PFree
/dev/sda2  cl lvm2 a--  <7,00g     0

```

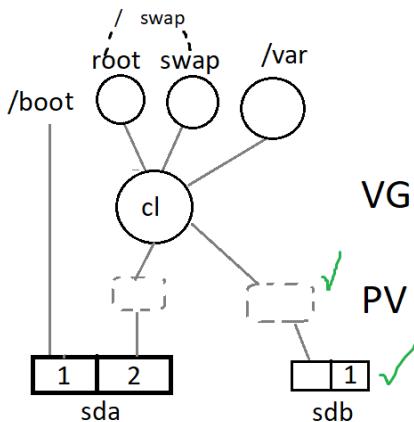
4. Nuestro siguiente paso es crear nuestro phisical volumen a partir del dispositivo sdb1. Ponemos “sudo pvcreate /dev/sdb1” -> pvcreate <dispositivo a partir del cual va a crear ese PV>

```
localhost ~]$ sudo pvcreate /dev/sdb1
Physical volume "/dev/sdb1" successfully created.
```

Comprobamos con pvs y pvdisplay si se ha creado.

```
localhost ~]$ sudo pvs
PV          VG Fmt Attr PSize PFree
/dev/sda2   cl lvm2 a-- <7,00g     0
/dev/sdb1    lvm2 --- <8,00g <8,00g
localhost ~]$
```

Ya tenemos nuestro phisical volumen listo.



5. Ahora tenemos que extender nuestro Volumen Group para que coja el nuevo phisical volumen.

Para monitorizar el estado de nuestro grupo de volúmenes hacemos prácticamente igual: “sudo vgdisplay” y “sudo vgs”

```
localhost ~]$ sudo vgs
VG #PV #LV #SN Attr  VSize  VFree
cl   1   2   0 wz--n- <7,00g     0
```

```
localhost ~]$ sudo vgdisplay
--- Volume group ---
VG Name           cl
System ID
Format           lvm2
Metadata Areas   1
Metadata Sequence No 3
VG Access        read/write
VG Status        resizable
MAX LV
Cur LV
Open LV
Max PV
Cur PV
Act PV
VG Size
PE Size
Total PE
Alloc PE / Size 1791 / <7,00 GiB
Free PE / Size   0 / 0
VG UUID          bF9AnX-7e1X-92sf-cfLn-Gzur-tWmy-fSLEsq
```



Mac
10 % dto | Estudiantes Profesores



iPad
6 % dto | Estudiantes Profesores



Rossellimac

Como vemos cl tiene 1 phisical volumen (/dev/sda2) y dos logical volumen (root y swap). Nada nuevo.

Para extender el tamaño del grupo de volúmenes tenemos vgextend. Ponemos “sudo vgextend”. Queremos extender el grupo de volúmenes. Tenemos que indicarle el grupo de volúmenes porque puede haber varios. Y tenemos que indicarle los phisical volumen. Como vamos a hacer la operación sobre un VG, nos acordamos que debemos ubicarlo primero y como sabemos que ese VG puede ser extendido en uno o mas PV, el resto de parámetros irán a la derecha.

En este caso es interesante saber comandos para detectar errores
“sudo vgextend cl /dev/sdb1”

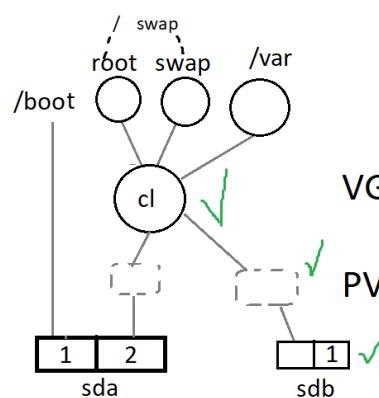
```
localhost ~]$ sudo vgextend cl /dev/sdb1
Volume group "cl" successfully extended
```

Ahora hacemos “sudo vgs” o “sudo vgdisplay” y veremos que ahora cl tiene 2 phisical volumen

```
localhost ~]$ sudo vgs
  [output]
  localhost ~]$ sudo vgdisplay
  [output]
  --- Volume group ---
  VG Name           cl
  System ID
  Format          lvm2
  Metadata Areas   2
  Metadata Sequence No 4
  VG Access        read/write
  VG Status        resizable
  MAX LV
  Cur LV
  Open LV
  Max PV
  Cur PV
  Act PV
  VG Size         14,99 GiB
  PE Size          4,00 MiB
  Total PE        3838
  Alloc PE / Size 1791 / <7,00 GiB
  Free  PE / Size 2047 / <8,00 GiB
  VG UUID          bF9AnX-7e1X-92sf -cfLn-Gzur-tWmy-fSLESq
```

Como podemos observar, ha aumentado la capacidad del VG y el espacio que tenia libre (antes no había espacio libre, ponía 0).

Ya tenemos el VG cl extendido con el PV.



6. Ahora vamos a crear el Logical Volumen.

Tenemos lvs para tener información de los volúmenes lógicos e igual con lvdisplay.

```
[root@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

[root@localhost ~]# sudo lvdisplay
--- Logical volume ---
LV Path          /dev/cl/swap
LV Name          swap
VG Name          cl
LV UUID          IgK7IH-d6jT-LwLC-SQm0-gSI7-XK1h-AK1doj
LV Write Access  read/write
LV Creation host, time localhost, 2021-09-30 06:02:31 -0400
LV Status        available
# open           2
LV Size          820,00 MiB
Current LE       205
Segments         1
Allocation       inherit
Read ahead sectors auto
- currently set to 8192
Block device    253:1

--- Logical volume ---
LV Path          /dev/cl/root
LV Name          root
VG Name          cl
LV UUID          8Y0XDJ-4kaS-p630-yBe9-c0hS-MIV9-c4cBy4
LV Write Access  read/write
LV Creation host, time localhost, 2021-09-30 06:02:31 -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
```

Dentro de los comandos de lv, tenemos lvcreate. Con este comando podemos crear una snapshot también a partir de un logical volumen -> interesante para backups. También se puede utilizar lv para configurar raid1. En estas prácticas utilziaremos los parámetros de asignarle un nombre y la longitud (como de largo queremos que sea). Por ultimo tenemos que especificarle el grupo de volúmenes VG donde queremos crear este volumen lógico. Ahora solo tenemos cl.

Sudo lvcreate -n new_var -L 3G cl

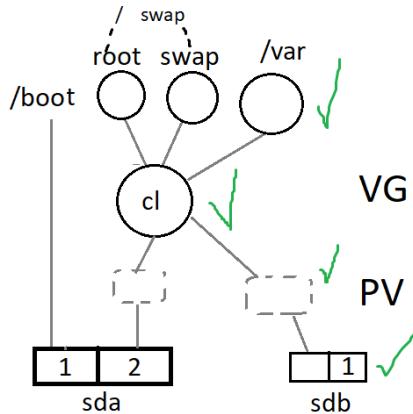
```
[root@localhost ~]# sudo lvcreate -n new_var -L 3G cl
  Logical volume "new_var" created.
```

Veamos si se ha creado bien con “sudo lvs” o “sudo lvdisplay”

```
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
```

```
--- Logical volume ---
LV Path          /dev/cl/new_var
LV Name          new_var
VG Name          cl
LV UUID          T18u6Z-IzgY-gduh-JgNn-dHwq-vyb4-Xs4kGb
LV Write Access  read/write
LV Creation host, time localhost.localdomain, 2021-09-30 17:12:50 -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
```

Ya tenemos nuestro volumen lógico preparado para new_var.



A nivel de configuración, lo que es la parte de gestión de LVM estaría lista.

Nos quedan cuestiones relativas a Sistemas Operativos. Profundizaremos en cuestiones como copiar archivos y esas cosas.

PARTE B

0. Crear un filesystem para un LVM
1. Tendremos que acceder al LV. (montarlo) para que el SO pueda acceder a él
2. Copiaremos la información de /var al Logical Volumen. Esta copia debemos hacerla de forma atómica.
3. Indicar al SO donde va /var
4. Liberar espacio

Pasos:

- En la terminal “`sudo mkdir /new_var`”. -> creamos el punto de montaje

Ahora utilizamos el comando `mount` - ¿Qué sistema de archivos tiene el volumen lógico que acabamos de crear? Lo necesitamos para montarlo (Se trata de montar un sistema de archivos). No lo sabemos porque no hemos creado ningún sistema de archivos. Por eso el paso 0.

Para crear el filesystem utilizaremos “`mkfs`”. Tendremos que especificar el tipo y el lugar donde lo vamos a crear. En nuestro caso en el volumen lógico.

Tenemos ahora distintas posibilidades. Tenemos que pensar. Al tener archivos grandes el mas adecuado es `xfs`. Para redimensionar los volúmenes lógicos a día de hoy es mas complejo. Por eso vamos a coger mejor `ext4`.

Ponemos “`sudo mkfs -t ext4 /dev/c1/new_var`”

```
[root@localhost ~]# sudo mkdir /new_var
[sudo] password for root
[root@localhost ~]# ls
lvdisplay
[root@localhost ~]# sudo mkfs -t ext4 /dev/c1/new_var
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: 3d24f830-b9c9-474c-a272-5b3011d8fa0d
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
```

- Una vez tenemos el sistema de archivos creados vamos a hacerlo accesible mediante el comando `mount`.

“`mount /dev/mapper/c1-new_var...`” -> una opción pero es mejor la de abajo

“`sudo mount/dev/c1/new_var /new_var`”

```
[root@localhost ~]# sudo mount /dev/c1/new_var /new_var/
[ 3319.022477] EXT4-fs (dm-2): mounted filesystem with ordered data mode. Opts: (null)
```

Vamos a asegurarnos con `mount` que está ahí con “`mount`”.

```
/dev/mapper/c1-new_var on /new_var type ext4 (rw,relatime,seclabel)
[root@localhost ~]# _
```

- Vamos a pasar a hacer la copia de los datos. Cuando hablamos de copia atómica nos referimos a evitar los cambios en la información mientras estamos copiandola. Podemos hacerlo mediante `snapshot` o simplemente evitando que los usuarios accedan a esta información.



Mac
10 % dto | Estudiantes Profesores



iPad
6 % dto | Estudiantes Profesores



Rossellimac

Para hacer esto utilizaremos el comando “systemctl”. Este comando controla Systemd. Esto es un trozo de software que rompe la filosofía de UNIX de hacer una sola cosa pero muy bien ya que hace varias cosas (controlar servicios, monitorizarlos, cron...). Tecleamos “sudo systemctl isolate rescue” y con esto conseguimos cambiar el modo de ejecución. Hay un bugg, nos sacará, nos logeamos como root y volvemos a hacerlo.

Si ponemos “systemctl status” podremos ver que ha cambiado el estado a “maintenance”.

```
[root@localhost ~]# systemctl status
● localhost.localdomain
  State: maintenance
    Jobs: 0 queued
   Failed: 0 units
     Since: Thu 2021-09-30 16:36:46 EDT; 1h 5min ago
   CGroup: /
           └─init.scope
```

- d. Como ya tenemos nuestro volumen lógico montado, vamos a pasar a copiar ahora si. Utilizaremos “cp” pero vamos a ver las opciones porque hay unos cuantos metadatos como los permisos y las fechas que nos podemos cargar al hacer la copia. Para preservar todos esos atributos/metadatos y para hacer una copia recursiva con los directorios podemos utilizar la opción -a o –archive:
`cp -a /var/. ./new_var/`

```
[root@localhost ~]# cp -a /var/. ./new_var/
[root@localhost ~]# _
```

Podemos hacer un “ls -laZ /var/.” para ver que las cosas siguen ahí. (-Z es para ver lo del contexto?) y “ls -laZ /new_var/”.

- e. Una vez que ya hemos copiado la información, le tenemos que decir al SO donde tiene que asignar el punto de montaje /var/. Para ello vamos a irnos al archivo del filesystemtable ubicado en /etc/.
`cd /etc/`
`less fstab`
Podemos encontrar ayuda sobre el archivo con “man fstab” para ver la información, los campos, etc.

Vamos a editar este archivo con vi (SABER COMANDOS DE VI). Utilizando la sintaxis del mapper vamos a introducir una nueva línea:

```
"/dev/mappet/cl-new_var /var ext4 defaults 0 0"
```

Los 0s corresponden a la prioridad que se le da en el caso de haber un backup

```

#
# /etc/fstab
# Created by anaconda on Thu Sep 30 06:02:32 2021
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
/dev/mapper/cl-root      /          xfs    defaults        0 0
UUID=93c7e900-4314-4801-97c6-cf367b6e11bd /boot
/dev/mapper/cl-swap      swap     swap    defaults        0 0
/dev/mapper/cl-new_var   /var     ext4   defaults        0 0

```

Como en C, por cada alloc hay que hacer un free. Aquí igual, por cada mount hay que hacer un umount. Si tecleamos “mount” vemos que new_var esta montado en new_var.

Tecleamos ahora “**umount /new_var**”

Y ahora “**mount -a**” para montar todos los archivos indicados en el fstab.

```
[root@localhost etc]# mount -a
[ 5025.114230] EXT4-fs (dm-2): mounted filesystem with ordered data mode. Opts: (null)
[ 5025.114230] Udevd[114]: 1 device added
```

Ahora hacemos lsblk para ver donde está montado y vemos que ahora tenemos nuestro /var montado en nuestro nuevo volumen lógico creado (cl-new_var). Antes de hacer “mount -a” esa parte de la columna de MOUNTPOINT estaba vacia en la fila de cl-new_var (aun no estaba montado)

```
[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    0  58,3M 0 rom
```

Así, tenemos la configuración deseada. Hasta cierto punto.

- Nos queda liberar espacio. Con esto nos referimos principalmente a: ¿qué ha pasado con ese /var que teníamos asignado originalmente? Recordemos que hemos hecho un cp. El /var de nuestro disco lo hemos copiado al /Var del nuevo volumen lógico. Está ocupando espacio en nuestro disco. Debemos liberar. -> Para esto vamos a ir marcha atrás.

Editamos el fstab otra vez, comentamos la línea que agregamos y hacemos un mount -a. Con esto, si hacemos lsblk vemos que sigue montado. Hacemos “umount /dev/cl/new_var” y ahora si, con lsblk vemos que no aparece el punto de montaje /var. Ahora mismo /var esta montado en var. Antes estaba montado en dos sitios.

Ahora descomentamos la línea de fstab. Para asegurarnos de que cuando lo llamemos esta preparado.

Ahora vamos a mover var a otra parte -> /var_old

“mv /var /var_old”

¿Con qué finalidad? Por si hay algún problemilla, poder recuperar la información a golpe de mover un directorio.

Volvemos a hacer “mount –a”.

¿Qué ocurre? Como hemos movido /var a /var_old, cuando llamamos a “mount -a” nos dice que el punto de montaje no existe. Lo creamos a mano -> “mkdir /var”

Si hacemos “ls / -laZ” -> el contexto de Linux para ese /var difiere del /var_old -> Para ello tenemos el comando “restorecon” que restaura el contexto de los directorios correspondientes.

Ponemos “restorecon /var”

```
ls -lZ /var_
object_r:sysfs_t:s0
object_r:tmp_t:s0
object_r:usr_t:s0
object_u:object_r:var_t:s0
object_r:var_t:s0
/var_
```

Ha cambiado el contexto a var_t

Y ahora podemos hacer mount -a

Con un lsblk vemos que tenemos nuestro var montado y si nos fijamos podemos hacer un ls / y tenemos los bloques de disco accesibles a través del directorio var_old. Cuando veamos que no nos sirven, podremos borrarlo y seguir funcionando con nuestro /var.

FIN

LECCIÓN 3

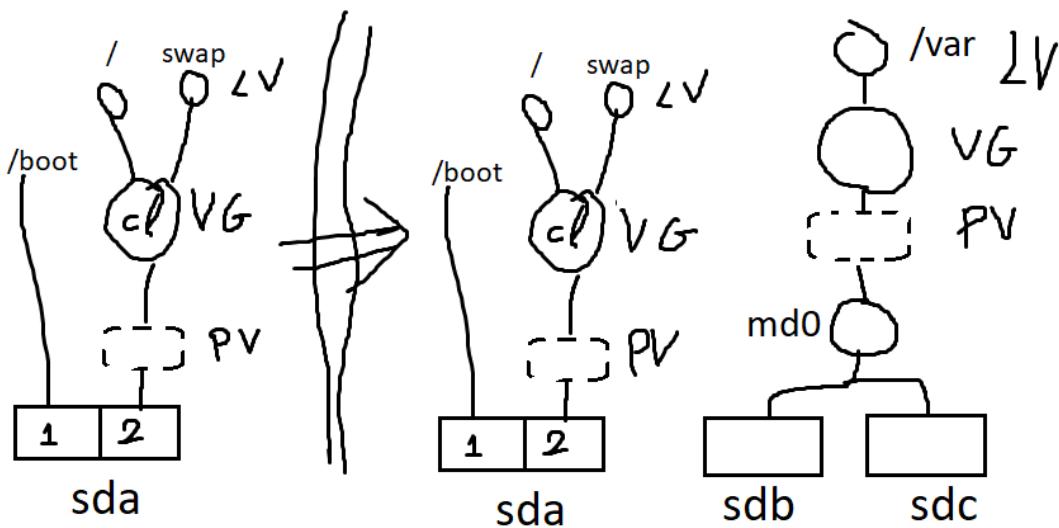
Un cliente quiere emular el servidor de la práctica anterior. Pero es más consciente de la legislación y nos pide explícitamente que cifremos la información. Además nos pide que esa información esté siempre disponible. Ya sabemos que si queremos disponibilidad tenemos que tener otra copia de la información: una configuración de RAID1 por software.

Partimos de una máquina de 0. Existe la posibilidad de reconvertir la máquina de la lección 2 para esto. Pero en esta lección partiremos de una configuración inicial. Creamos la máquina virtual con centOS como siempre.

La instalación del centOS es exactamente igual que en la Lección 2.

Queremos añadir a la configuración “actual” el /var nuevo como en la práctica anterior. Pero queremos que esté cifrado. Tenemos que añadir como mínimo dos discos (sdb y sdc para RAID1) y a partir de ellos crear el md0 (multidivice). Sobre este multidivice configurado como RAID1 crearemos nuestro PV y tendremos dos opciones:

1. Extender el grupo de volúmenes cl con nuestro nuevo PV. (como la semana pasada) y a partir de ahí crear el LV para /var. Pero no tendríamos la certeza de que lo que estuviéramos escribiendo en /var va a ir a parar a nuestro RAID1. En esta ocasión si ocurriría porque el espacio de sda está completo y lógicamente la nueva información va a meterse en los dispositivos RAID. Pero podría no ser el caso y podríamos tener espacio libre en cl. No tenemos una certeza.
2. La solución de diseño más correcta se basa en crear un nuevo VG y a partir de ese VG ya podemos meter todos los LV que queremos que vayan en la configuración de RAID1. (p.e otro caso es que a lo mejor nos interesa por prestaciones tener unos LV en unos dispositivos más rápidos. Crearíamos un grupo de volúmenes con esos dispositivos más rápidos para esos LV que necesitan más velocidad).





Mac

10 % | Estudiantes
dto Profesores



iPad

6 % | Estudiantes
dto Profesores

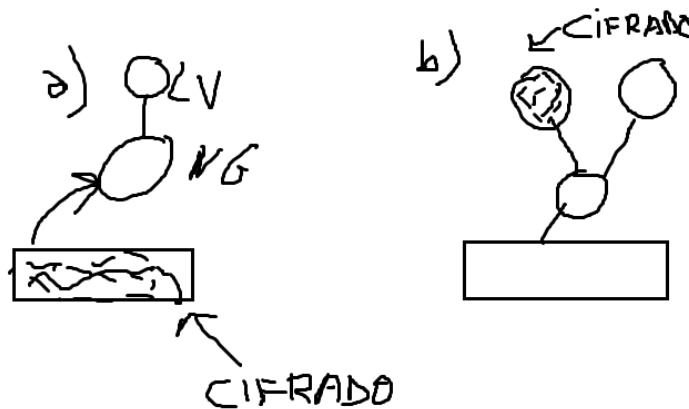


Rossellimac[®]

Además, nos indica que /Var tiene que estar cifrado. Hay que cifrar la información. Se nos presentan también dos posibilidades:

- LVM sobre LUKS: Vamos a tener un cifrado de disco.
 - Se cifra todo el disco
 - Una vez activado es formateado
 - Se crea el VG
 - Se crea el LV
- LUKS sobre LVM: Utilizaremos este
 - Utilizamos LVM
 - El LV es lo que se cifra

Ventaja: podemos tener otro LV que no nos hace falta cifrarlo.



LUKS: Linux Unified Key Setup. es una especificación de cifrado de disco. Su particularidad es que especifica un formato estándar en disco, independiente de plataforma

FDE (Full Disk Encryption): método para encriptar discos duros de tal manera que todos los datos en la unidad estén siempre encriptados, sin el uso de soluciones de encriptación de terceros. En ninguno de estos esquemas estamos alcanzando FDE por el problema de GRUP y /boot.

Una vez creada la Máquina Virtual le añadimos los dos discos: sdb y sdc. Le vamos a dar 2GB a cada uno esta vez

Una vez creados los dos discos nuevos, iniciamos la máquina.

Lo primero que haremos es comprobar que nuestra máquina tiene los dos discos nuevos.

Si están:

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	
sdc	8:32	0	2G	0	disk	
sr0	11:0	1	1024M	0	rom	

Ahora tenemos que crear el RAID1. Una de las novedades de esta lección es el comando mdadm (multidivice administration).

Como buena práctica, vamos a formatear. Vamos a crear las particiones correspondientes dejándole el espacio automáticamente con fdisk y vamos a crear las particiones para sdb y sdc con fdisk.

- “sudo fdisk /dev/sdb”
- Nos logeamos
- “p” -> mostrar particiones

```
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: 0x849f6797
```

- “n” -> creamos nueva partición
- “p” -> primaria
- Dejamos el tamaño predeterminado.
- “p” -> comprobamos que esta correcto

```
Orden (m para obtener ayuda): n
Tipo de partición
  p  primaria (0 primaria(s), 0 extendida(s), 4 libre(s))
  e  extendida (contenedora 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/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: 0xc2ec3941
```

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

- “w” -> escribimos

Hacemos lo mismo con “sudo fdisk /dev/sdc”

```
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: 0x7d2dfbc4

Orden (m para obtener ayuda): n
Tipo de partición
  p  primaria (0 primaria(s), 0 extendida(s), 4 libre(s))
  e  extendida (contenedora 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):

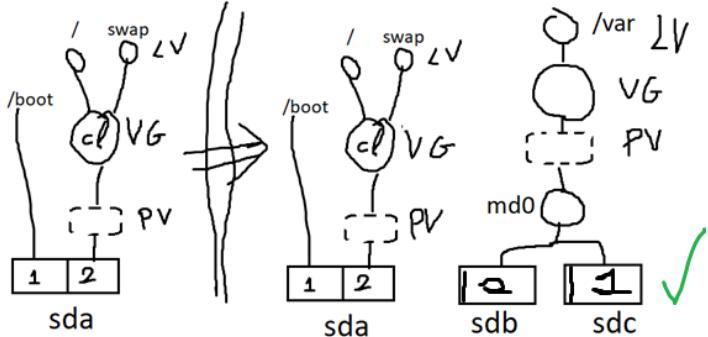
Crea una nueva partición 1 de tipo 'Linux' y de tamaño 2 GiB.

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: 0x7d2dfbc4

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

Comprobamos que se han creado bien con “lsblk”

```
alhost ~]$ lsblk
-bash: lsblk: no se encontró la orden
alhost ~]$ 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
└─c1-root 253:0  0  6,2G  0 lvm /
└─c1-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
```



Ya podemos pasar a crear nuestro multidivice, nuestro RAID1.

Para definir nuestro RAID1 vemos como en las opciones de mdadm tiene el siguiente formato:

“mdadm <mode> <raiddevice> <options> <component-devices>

Mode: modo en el que estamos operando (crear, monitorizar...)

raiddevice: dispositivo con el que queremos operar (nos inventamos el nombre)

Option: las distintas opciones

<component-devices>: ubicaremos los distintos dispositivos

Vamos a crear nuestro dispositivo multidivice con:

“sudo mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1”

--level: nivel de raid

--raid-devices: numero de dispositivos en nuestro raid1

```
alhost ~]$ sudo mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1
[sudo] password for
Sorry, try again.
[sudo] password for
Sorry, try again.
[sudo] password for
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?
```

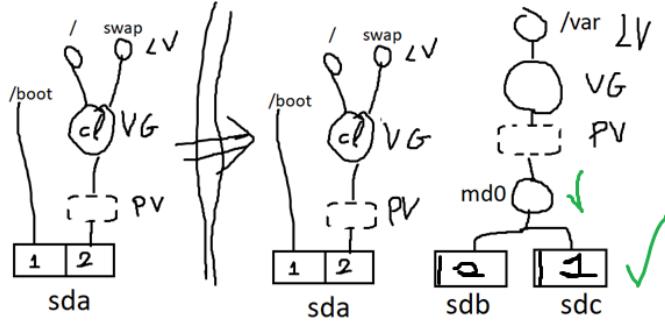
Como vemos, nos indica que no vamos a dejar espacio suficiente para los metadatos en caso de que queramos utilizarlo como disco de arranque pero nos da igual porque lo tenemos particionado y, a parte, /Var no va a ser utilizado como disco de arranque.. Continuamos poniendo “y”.

Ya tenemos nuestro dispositivo creado.

Hacemos “ls /dev” y ya veremos nuestro md0.

```
alhost ~]$ ls /dev
autofs      fuse
block      hpet
bsg        hugepages
bus        hwmon
cdrom      initctl
char       input
c1         kms
console    log
core       loop-control
cpu        mapper
cpu_dma_latency  mcelog
disk      md0
fd0
```

WUOLAH



Ahora vamos a utilizar nuestro comando “pvcreate” para crear el PV (physical volumen) a partir de nuestro md0.

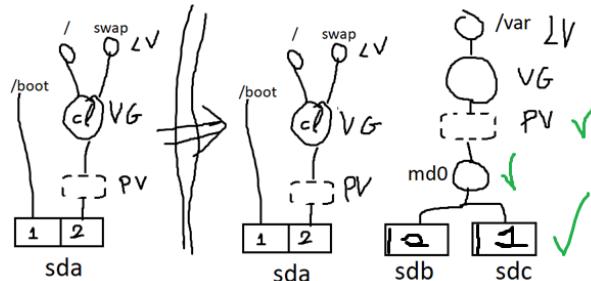
Hacemos “sudo pvs” y vemos que tenemos nuestro VG cl utilizando el PV /dev/sda

```
localhost ~]# sudo pvs
[sudo] password for sergiohc:
PV          VG Fmt Attr PSize  PFree
/dev/sda2   cl lvm2 a-- <7,00g    0
```

Hacemos “sudo pvcreate /dev/md0”. Lo creamos y comprobamos.

```
localhost ~]# sudo pvcreate /dev/md0
Physical volume "/dev/md0" successfully created.
[sergiohc@localhost ~]# sudo pvs
PV          VG Fmt Attr PSize  PFree
/dev/md0    lvm2 --- <2,00g <2,00g
/dev/sda2   cl lvm2 a-- <7,00g    0
```

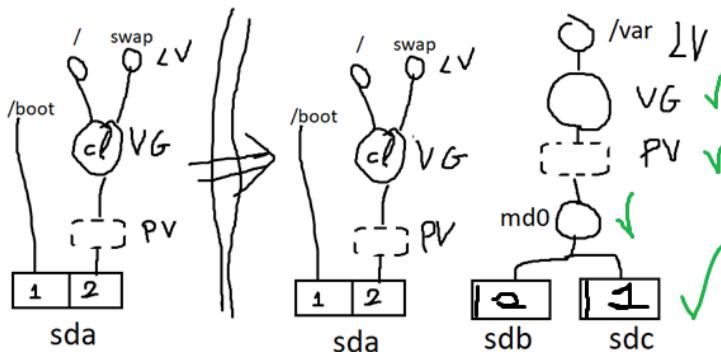
Ya tenemos nuestro PV. Como vemos aun no tiene un VG.



Para crear nuestro Volumen Group ponemos “sudo vgcreate vg_raid1 /dev/md0”. Comprobamos con “sudo vgs”

```
[calhost ~]# sudo vgcreate vg_raid1 /dev/md0
Volume group "vg_raid1" successfully created
[calhost ~]# sudo vgs
VG #PV #LV #SN Attr  VSize  VFree
cl     1   2   0  wz--n- <7,00g    0
vg_raid1 1   0   0  wz--n- <2,00g <2,00g
```

Como vemos, tenemos cl con 2 LV y vg_raid1 con 1PV pero sin LV.





Rossellimac®



Mac

10 % dto | Estudiantes Profesores



iPad

6 % dto | Estudiantes Profesores

Vamos a crear nuestro LV (/var).

Vamos a echar un vistazo a los volúmenes lógicos con “sudo lvs” y vamos a crear nuestro volumen lógico con “sudo lvcreate -n new_var -L 1.8G vg_raid1”

```
calhost ~]$ 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
          alhost ~]$ sudo lvcreate -n new_var -L 1.8G vg_raid1
  Rounding up size to full physical extent 1,88 GiB
  Logical volume "new var" created.
  calhost ~]$ 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
```

Ya vemos

como tenemos nuestro LV new_var creado dentro del VG vg_raid1 con un espacio de 1.8G.

Ahora tenemos que pasar a hacer los pasos similares de la pagina anterior (traspaso de información y eso). La parte de LVM ya la hemos hecho.

Pero antes tenemos que cifrar el volumen lógico creado. Utilizaremos la herramienta “cryptsetup”.

La tendremos que instalar con dmf. (dmf es el remplazo de yum)

Necesitamos acceso a internet para poder instalarlo. Lo comprobamos con “ip addr” para ver si tenemos una IP asignada o hacemos ping a Google o a otra web para ver si se hace bien.

```
ocalhost ~]$ 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
    inetc6 ::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:fd:d7:cd brd ff:ff:ff:ff:ff:ff
ocalhost ~]$ ping www.google.com
ping: www.google.com: Name or service not known
```

En mi caso, no tengo. Por eso, levantamos la interfaz enp0s3:

Ponemos “sudo ifup enp0s3”

```
ocalhost ~]$ sudo ifup enp0s3
[sudo] password for
Conexión activada con éxito (ruta activa D-Bus: /org/freedesktop/NetworkManager/ActiveConnection/1)
```

Ahora ya si podemos ver que tenemos conexión:

```
ocalhost ~]$ 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
    inetc6 ::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:fd:d7:cd 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 86374sec preferred_lft 86374sec
    inetc6 fe80::cd:4a25:5be2:393e/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
ocalhost ~]$ ping www.google.com
PING www.google.com (142.250.185.4) 56(84) bytes of data.
64 bytes from mad41s11-in-f4.1e100.net (142.250.185.4): icmp_seq=1 ttl=116 time=15.5 ms
64 bytes from mad41s11-in-f4.1e100.net (142.250.185.4): icmp_seq=2 ttl=116 time=17.3 ms
```

Ya podemos hacer “dnf search cryptsetup” para hacer la búsqueda.

```
[root@localhost ~]# dnf search cryptsetup
CentOS-8 - AppStream
CentOS-8 - Base
CentOS-8 - Extras
=====
===== Coincidencia exacta en Nombre: cryptsetup =====
cryptsetup.x86_64 : A utility for setting up encrypted disks
=====
===== Coincidencia en Nombre , Resumen: cryptsetup =====
cryptsetup-libs.i686 : Cryptsetup shared library
cryptsetup-libs.x86_64 : Cryptsetup shared library
=====
===== Coincidencia en Nombre: cryptsetup =====
cryptsetup-devel.i686 : Headers and Libraries for using encrypted file systems
cryptsetup-devel.x86_64 : Headers and libraries for using encrypted file systems
cryptsetup-reenrypt.x86_64 : A utility for offline reencryption of LUKS encrypted disks.
```

Ahí tenemos las herramientas.

Ponemos “sudo dnf install cryptsetup -y” (el -y es para que no nos pregunte)

Una vez instalado, hablemos de la sintaxis del comando:

Tendremos distintas acciones: open, format, etc. Esta acción siempre va a venir precedida del prefijo “luks”: luksopen, luksformat...

Vamos a formatear el volumen lógico que hemos creado. Vamos a darle formato. Vamos a cifrarlo. Para ello ponemos: “sudo cryptsetup luksFormat /dev/vg_raid1/new_var”

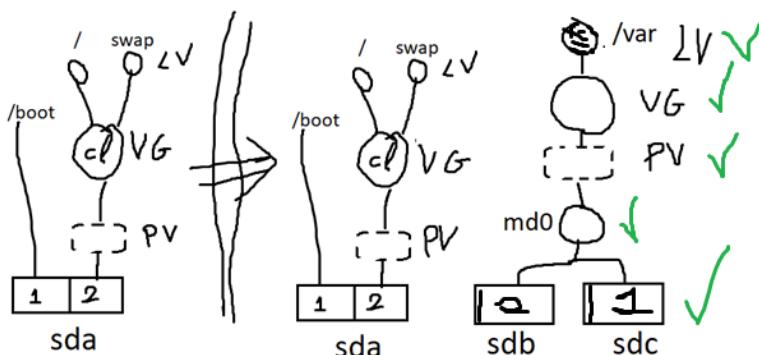
Ahora nos da un mensaje de advertencia. Nos esta preguntando si estamos seguros. Hay que escribir “YES”.

Ahora nos preguntan por la contraseña. Ponemos la de siempre.

```
[root@localhost ~]# sudo cryptsetup luksFormat /dev/vg_raid1/new_var
=====
WARNING!
=====
Esto sobreescribirá 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:
```

Con esto ya tendríamos cifrado el volumen. Antes de cifrar el volumen, la documentación de redhat nos indica que es bueno utilizar el comando “shred”. Esto lo que hace es generar basura para dar un extra de seguridad. Así, se parte de una situación aleatoria que nadie pueda partir de antemano.



Una vez que ya tenemos cifrado el disco, tenemos que acceder a él. Para ello, tenemos que volver a hacer uso de cryptsetup:

Ponemos “sudo cryptsetup luksOpen /dev/vg_raid1/new_var vg_raid1-new_var_crypt” (vg_raid1-new_var_crypt es el nombre. Le tenemos que pasar un nombre para hacer referencia a ese volumen “activado” -> al abrir se ““activa””). Nos pide la frase de paso (la contraseña puesta antes)

```
localhost ~]$ sudo cryptsetup luksOpen /dev/vg_raid1/new_var vg_raid1-new_var_crypt
[[6~[sudo] password for
Sorry, try again.
[sudo] password for
Introduzca la frase contraseña de /dev/vg_raid1/new_var:
|      localhost ~]$ _
```

Ahora tenemos una cierta incertidumbre para saber si ha funcionado o no.

Ponemos “ls /dev/mapper/”

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

Vemos como tenemos nuestro volumen cifrado (vg_raid1-new_var) y tenemos accesible nuestro volumen cifrado activado (vg_raid1-new_var_crypt)

Ya tenemos nuestro diseño de la infraestructura lista. A partir de aquí hay que repetir los pasos que hicimos para montar /var en un volumen lógico (lo que hicimos en la lección 2).

Después, habrá que hacer un paso final para indicar al SO cuando arranque que tiene que activar el volumen cifrado, es decir, que ejecute automáticamente “sudo cryptsetup luksOpen /dev/vg_raid1/new_var vg_raid1-new_var_crypt”. Hay un archivo especial para ello.

Ahora pasamos a crear nuestro sistema de archivos

“sudo mkfs -t xfs /dv/mapper/vg_raid1-new_var_crypt”

(recordemos que tenemos que decir que lo cree en el volumen activado, si no no va a ser capaz de crear el sistema de archivos)

```
[root@localhost ~]# sudo mkfs -t xfs /dev/mapper/vg_raid1-new_var_crypt
meta-data=/dev/mapper/vg_raid1-new_var_crypt isize=512    agcount=4, agsize=116992 blks
          =                      sectsz=512   attr=2, projid32bit=1
          =                      crc=1     finobt=1, sparse=1, rmapbt=0
          =                      reflink=1
data     =                      bsize=4096   blocks=467968, imaxpct=25
          =                      sunit=0     swidth=0 blks
naming   =version 2           bsize=4096   ascii-ci=0, ftype=1
log      =internal log        bsize=4096   blocks=2560, version=2
          =                      sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                extsz=4096   blocks=0, rtextents=0
```

Ponemos “sudo systemctl isolate rescue.target” -> Volvemos a tener el bugg y repetimos Luego ponemos “systemctl status” para ver si se ha hecho bien.

```
[root@localhost ~]# systemctl status
● localhost.localdomain
  State: maintenance
    Jobs: 0 queued
  Failed: 0 units
   Since: Thu 2021-10-07 18:17:24 EDT; 32min ago
```

WUOLAH

Ahora creamos el punto de montaje con "mkdir /new_var"

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

Montamos el volumen activado en new var:

- Ponemos: "mount /dev/mapper/vg_raid1-new_var_crypt /new_var"

```
[root@localhost ~]# mount /dev/mapper/vg_raid1-new_var_crypt /new_var/
[ 2390.909844] XFS (dm-3): Mounting U5 Filesystem
[ 2390.933647] XFS (dm-3): Ending clean mount
```

Ahora copiamos con "cp -a /var/. /new_var/" y comprobamos si se ha copiado bien con el comando "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 oct  6 18:10 .
dr-xr-xr-x. 18 root root system_u:object_r:root_t:s0    239 oct  7 18:56 ..
drwxr-xr-x.  2 root root system_u:object_r:acct_data_t:s0  19 oct  6 17:49 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 oct  6 17:49 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 oct  6 17:49 db
drwxr-xr-x.  3 root root system_u:object_r:var_t:s0     18 oct  6 17:48 empty
drwxr-xr-x.  2 root root system_u:object_r:public_content_t:s0  6 may 10 2019 ftp
```

Ahora indicamos en fstab donde queremos que monte. Le vamos a poner
/dev/mapper/vg_raid1-new_var_crypt /var xfs default 0 0

Guardamos y salimos.

Ahora tenemos que indicarle al SO que cuando arranque active ese Volumen Lógico. Esto se hace con el archivo crypttab.

FORMATO: "nombre dispositivo encriptado contraseña opciones"

Le debemos especificar el dispositivo encriptado con el UID -> lo obtenemos con cryptsetup y con blkid (mejor esta ultima)

```
[root@localhost ~]# blkid
/dev/sda1: UUID="37fdfa08-df6b-4558-a458-7d3933a0550e" TYPE="ext4" PARTUUID="8e9f8efd-01"
/dev/sda2: UUID="6CmSW4-ZZFi-Fxza-oPHR-nD3i-0vYk-2bmNnG" TYPE="LVM2_member" PARTUUID="8e9f8efd-02"
/dev/sdb1: UUID="b07d9ad3-1bb7-0a41-534c-ea4fd93d1303" UUID_SUB="d3fac4d4-9cf7-2fa8-d0ae-fa4a26a9999e" LABEL="localhost.localdomain:0" TYPE="linux_raid_member" PARTUUID="c2ec3941-01"
/dev/sdc1: UUID="b07d9ad3-1bb7-0a41-534c-ea4fd93d1303" UUID_SUB="678d2889-6e14-82fa-23a1-aa5ab85464f6" LABEL="localhost.localdomain:0" TYPE="linux_raid_member" PARTUUID="7d2dfbc4-01"
/dev/mapper/cl-root: UUID="e88d8390-61c1-48df-bd78-bb5d66d382da" TYPE="xfs"
/dev/mapper/cl-swap: UUID="37b7dc63-9b7f-4b60-86c1-370d892b3473" TYPE="swap"
/dev/md0: UUID="HeHcRf-6RsA-gmK4-HUQK-SyIi-tWik-xFJFST" TYPE="LVM2_member"
/dev/mapper/vg_raid1-new_var: UUID="c399cdf6-844f-4296-b6e5-06ffdcc251593" TYPE="crypto_LUKS"
/dev/mapper/vg_raid1-new_var_crypt: UUID="33370fffd-3cd6-42df-99c2-f19261c4c024" TYPE="xfs"
```



Rossellimac

Mac
10 % dto | Estudiantes Profesores



iPad
6 % dto | Estudiantes Profesores

Como vemos, es un lio. Vamos a filtrar la salida y nos vamos a quedar con los que estan cifrados. Ponemos “blkid | grep crypt”

```
[root@localhost ~]# blkid | grep crypt
/dev/mapper/vg_raid1-new_var: UUID="c399cdf6-844f-4296-b6e5-06ffdc251593" TYPE="crypto_LUKS"
/dev/mapper/vg_raid1-new_var_crypt: UUID="33370ffd-3cd6-42df-99c2-f19261c4c024" TYPE="ext4"
```

CUIDADO! Tenemos dos dispositivos. Queremos hacer referencia al volumen que está cifrado con crypto-LUKS.

Vamos a filtrarlo un poco mas

```
[root@localhost ~]# blkid | grep crypto
/dev/mapper/vg_raid1-new_var: UUID="c399cdf6-844f-4296-b6e5-06ffdc251593" TYPE="crypto_LUKS"
```

Ahora cogemos esta salida y la redirigimos a /etc/crypttab (este archivo aun no existe)

Ponemos por tanto: “blkid | grep crypto > /etc/crypttab”

```
[root@localhost ~]# blkid | grep crypto > /etc/crypttab
[root@localhost ~]#
```

Ahora editamos este archivo con “vi /etc/crypttab” y vamos a quitarle el prefijo porque no nos hace falta. También vamos a decirle que lo denominé “new_var_crypt” -> ES DECIR, UNA VEZ ACTIVADO LE AÑADIMOS EL SUFIJO CRYPT.

Por tanto, en primer lugar vamos a poner el nombre del volumen lógico ACTIVADO. Luego necesitamos el UID del volumen lógico cifrado sin activar. Le quitamos las comillas. Quitamos lo ultimo (TYPE) y le indicamos al final que no utilice ningún archivo extra con “none”

ANTES:

```
/dev/mapper/vg_raid1-new_var: UUID="c399cdf6-844f-4296-b6e5-06ffdc251593" TYPE="crypto_LUKS"
```

DESPUES

```
vg_raid1-new_var_crypt UUID=c399cdf6-844f-4296-b6e5-06ffdc251593 none
```

Ahora vamos a probar a hacer un reboot y debería estar todo.

NO HEMOS HECHO LO DE LIBERAR ESPACIO PERO LO PODRÍAMOS HACER:

```
/etc/crypttab: 1, 180 w/1tch
[root@localhost ~]# mv /var/ /var_old
[root@localhost ~]# mkdir /var
[root@localhost ~]# restorecon /var
```

Hacemos reboot.

```
Please enter passphrase for disk vg_raid1-new_var (vg_raid1-new_var_crypt) on /var/! [ 6.488580]
EXT4-fs (sda1): mounted filesystem with ordered data mode. Opts: (null)
```

Aquí vemos como está arrancando y efectivamente el crypttab le esta diciendo al SO que active el volumen lógico new_var y cuando lo active lo va a denominar vg_raid1-new_var_crypt y ahí va a montar /var.

Tecleamos la contraseña .

Activa el volumen si todo va bien y si no ha habido ningún problema, montará /var en ese volumen lógico.

Ponemos “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	

Efectivamente vemos que tenemos /var montado en un volumen cifrado que está en raid1 con los dos discos sdb y sdc.

Ya tenemos también nuestro /var_old para liberarlo cuando sea necesario.

Una cuestión importante: si nos fijamos hablamos de cifrado pero ahora mismo el volumen para swap no esta cifrado. -> ROJO -> Toda la información que tengamos en /var podría pasar en algún momento en /var y de RAM al volumen swap. Alguien con mala intención podría tener acceso a esa información.

Para solucionar esto podríamos hacer uso del comando “swapoff” y desactivar el swap para poder decir con total seguridad que esta configuración es correcta en términos de seguridad y cumplimiento de la ley.

1. Instalación de centOS
2. Añadir dos discos → para crear el RAID1 →
3. Crear un PV
4. Crear un VG
5. Crear un LV
6. Definir LUKS sobre el VL → utilizaremos cryptsetup. Tendremos que definir una entrada en un fichero llamado /etc/crypttab → crypttab → igual que fstab pero donde vamos a cargar los volúmenes cifrados
7. Definimos sistema de ficheros.
8. Igual que la semana pasada → copiar el contenido de /var/ al nuevo volumen /new_var

Examen:

1h30min

Ejercicio practico → partiremos de una hipótesis como los ejercicios

history -c

history | more

history | less

prompt