PRÁCTICA 1: Virtualización e instalación de sistemas operativos

Usaremos los SOs UbuntuServer(Canonical) modo gráfico y CentOS (RedHat), sin interfaz gráfica (también tiene interfaz gráfica el instalador).

Lección1: Instalación y configuración de UbuntuServer.

Lección2: Configuración de LVM con CentOS.

Lección3: Configuración de RAID1 con CentOS.

Diferencia entre máquina virtual y un contenedor:

-(proceso)Una máquina virtual simula el hardware entero de un PC para poder instalarle encima un SO. Da la posibilidad de montarte sistemas de forma fácil sin tener que recurrir a pagarlos, debido a que no hará falta comprar equipo nuevo. Además, con total libertad de poder conectarlos entre ellos, tener total operatividad y por supuesto salida a internet. No tienen comunicación entre sí a no ser que le emulemos las interfaces de red.

-(hebra)Un contenedor es, al igual que una máquina virtual emula el hardware entero de una máquina, el contenedor usa el propio hardware del ordenador donde está (usa el hardware de tu propio host, de tu propio equipo, para meterle encima la emulación de un software). Emula una interfaz para que te abstraiga del SO usando el tuyo, es decir, usa el tuyo y mediante una interfaz se abstrae de todos los demás, para poder montar encima el software emulado. Como Docker. Fue pensado idealmente para micro servicios, ya que no gasta recursos hasta que se necesita algo y se levanta el Docker. La utilidad se ha extendido más allá, cargando en un Docker sistemas operativos Ubuntu completos virtualizados. De forma que desde cualquier lado tu puedes encender una imagen de Docker de Ubuntu y el sistema operativo se te abre, sin tener que emular el hardware entero. A diferencia de las MV, el contenedor, al acceder a los mismos recursos que el ordenador anfitrión, se pueden compartir datos (recursos) entre ellos. Es decir, se enciende, sirve y apaga (lo levantas un momento, hace algo, te sirve algo, te devuelve algo y se vuelve a cerrar).

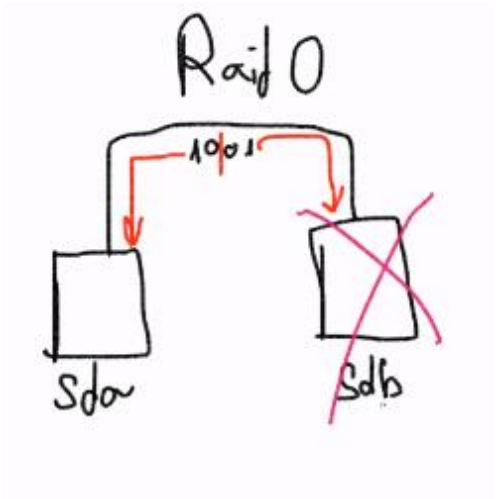
Una posible analogía es la diferencia entre proceso y hebra, dos máquinas virtualizadas completamente serian como dos procesos mientras que los contenedores serian como las hebras (que comparten “cosas”).

-***Un servidor*** es, una máquina que te sirve un servicio (pag.Web, imagen...). Idóneamente sin carga gráfica, como CentOS, que pertenece actualmente a RedHat, siendo de los más potentes. Windows a escala comercial es buen servidor, pero no es de código abierto y de pago. Suelen estar montados en una sala llena de clústeres con PCs montados en rack y se suele acceder a ellos de forma remota por *ssh*.

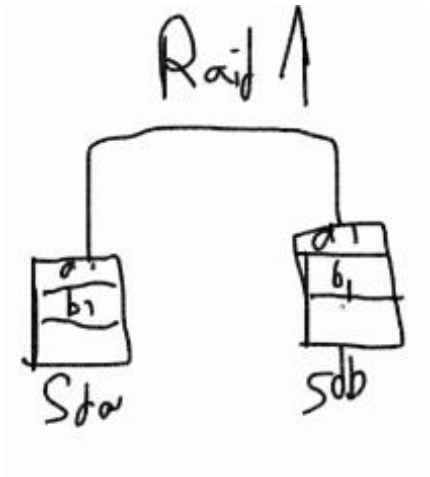
5 tipos de RAID (Redundant Array of Independent Disks) --> Sistema de redundancia que evita la perdida de datos. Todo lo que se escribe en un disco duro se escribirá automáticamente en otro. Completamente necesario para pequeñas y medianas empresas por si ocurre una catástrofe, no perder la información. Respaldo de información actualizado, dado que "todo lo que se escribe en un disco duro automáticamente se escribe en el otro/otros" aplicando diferentes tecnologías:

Tipos RAID (+ importantes):

+RAID0 - Es el más básico que existe. Necesita un mínimo de dos discos duros conectados entre ellos. Peculiaridad: si queremos guardar una palabra (1001) la parte por la mitad, ganando el doble de lecturas y el doble de escrituras respecto a velocidad. Problema: Si uno falla pierde toda la información porque realmente parte la información:



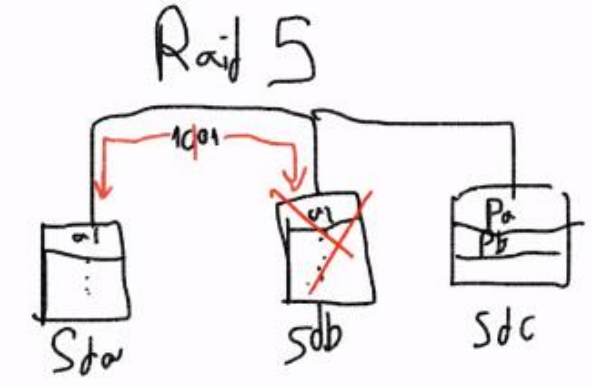
+RAID1 - Usado en la asignatura, el más útil a nivel básico, si los datos no son muy críticos, para uso doméstico y pequeñas medianas empresas. Necesita un mínimo de dos discos duros conectados entre ellos. Al contrario que el anterior, este si duplica la información.



Si falla uno de los discos duros, el sistema no arrancará porque te dirá que le falta un disco duro, pero si colocamos otro, los datos estarán dado que se hace la copia en tiempo real cuando arranca y el sistema vuelve a arrancar correctamente.

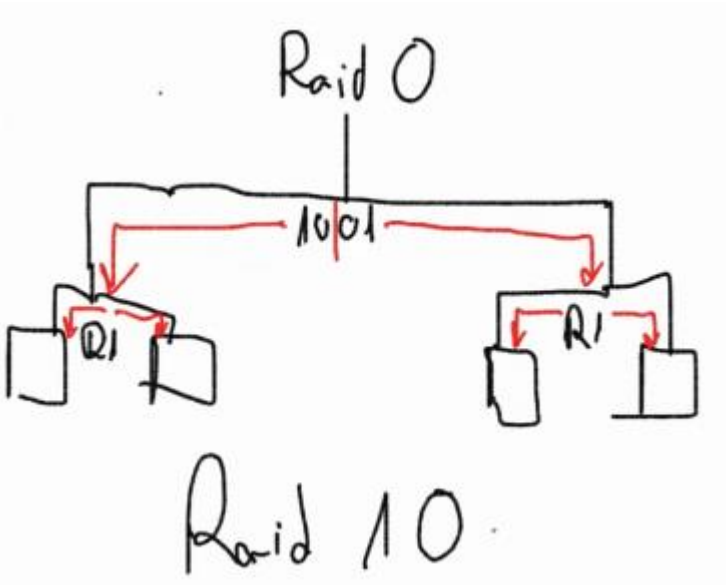
+RAID5 - Es lo que se debe montar en vez de un RAID0. Es como el 0 pero te permite recuperar la información. Consta de mínimo 3 discos duros. A priori, cuando te llega una palabra funcionará como el RAID0. En el tercer disco guardará bits de paridad para poder recuperar la información si se estropea sda o sdb, con el de paridad podremos calcular en tiempo real los bits que faltan a través de una serie de operaciones internas con los bits de paridad.

Si se estropea el de paridad, cámbialo rápido, ya que tu sistema RAID se habrá convertido en el RAID0). En el momento que enchufes el nuevo disco para los bits de paridad se actualizará cuando arranca el sistema.



+RAID6 - Es un RAID5 con dos discos duros de paridad.

+RAID10 - Combina la velocidad del RAID0 de lectura y escritura y la redundancia del RAID1 gracias a su arquitectura. Los más robustos con la arquitectura que incorpora un RAID0 en la capa superior y dos RAID1 en la inferior, de forma que el RAID0 parte la palabra y se lleva cada mitad a un RAID1 (metiéndose así cada mitad de la palabra en dos discos duros). Por tanto, es más útil a niveles muy elevados.



Diferencia entre “RAID software” y “RAID hardware”:

A parte de que sea más caro, usar un hardware dedicado siempre conlleva mayor velocidad y prestaciones y realmente, por software si se rompe el software del host de las máquinas virtuales, si es verdad que pierdes toda la información, aunque es el económicamente accesible a nivel usuario y de empresa común, aportando flexibilidad y bajo coste.

Nosotros montaremos un RAID software (internamente Ubuntu te da la manera de hacer ese RAID, consumiendo ciclos de CPU para hacerlo internamente, pero es muy barata ya que basta con comprar disco duro). En cuanto a velocidad y prestaciones es mejor el RAID por hardware, pero menos modificable que el RAID software ya que este último lo puedes modificar a tu gusto cambiando el tipo de RAID, pero si te compras hardware para, por ejemplo, RAID1, solo tendrás ese y si quieres otro será más caro.

RAID hardware: velocidad, prestaciones y coste alto.

RAID software: flexibilidad y bajo coste.

*NOTA:

Debemos tener encriptados todos los datos sensibles cuando gestionamos cosas de este tipo, por ley de protección de datos. Si no encriptamos los discos duros y se filtran los datos, el marrón es tuyo.

Haremos partición para el arranque del sistema, sacándolo fuera para que se pueda arrancar el sistema, y el resto del disco duro debe ir cifrado. Conlleva que al arrancar el sistema te pide la contraseña para descifrarlo.

LVM o Logical Volume Manager --> Te ayuda a flexibilizar los directorios del sistema, sobre todo los importantes. LVM te crea unas capas de abstracción de esas carpetas importantes del sistema para que, si en algún momento te quedas sin espacio o necesitas redistribuirlo, directamente coges ese volumen lógico y lo apuntas a un disco duro nuevo. Es una forma muy fácil de sin tener que echar todo el sistema abajo poder redimensionar una parte del sistema.

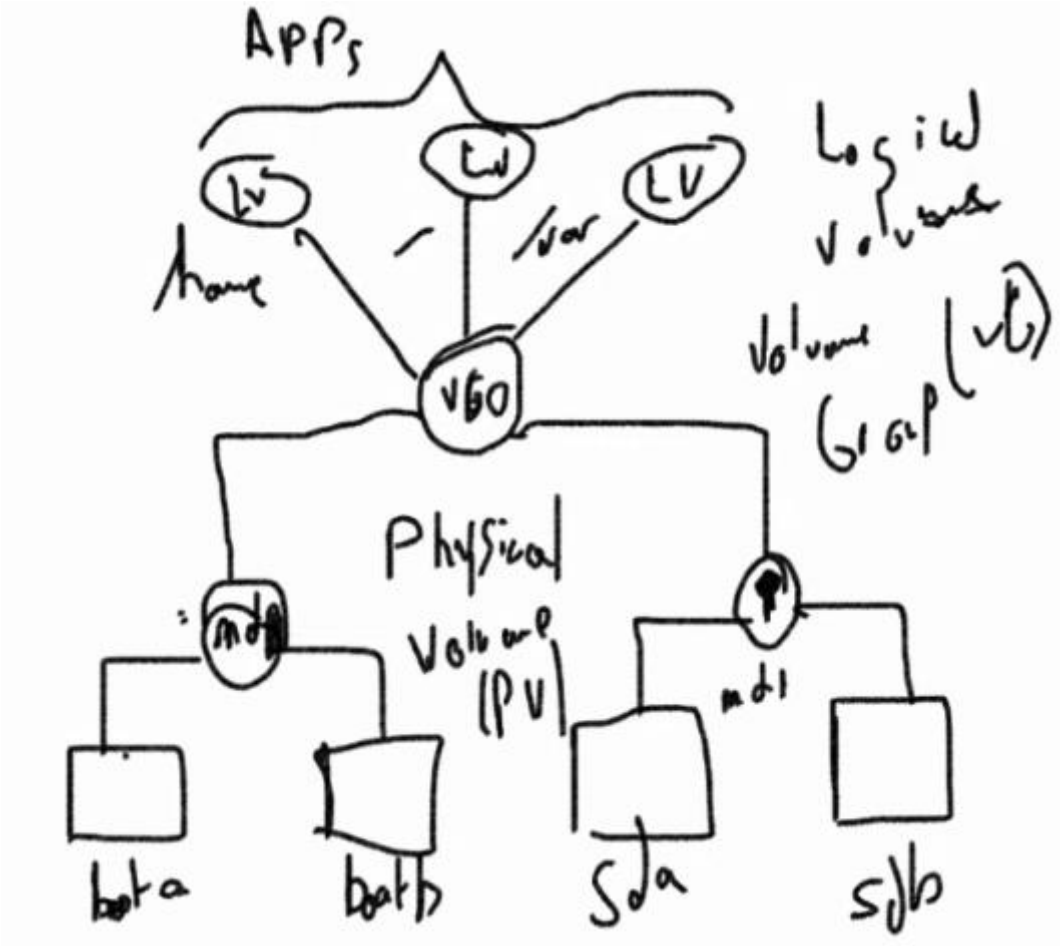
LECCIÓN1: Instalación y configuración de UbuntuServer

Anexo I: Contextualización de Escenarios

2.3. Lección 1

Usted está trabajando en una empresa proveedora de servicios y recibe la solicitud de un cliente que sea tener un servidor para la implantación de un comercio electrónico mediante un CMS.
Sin tener más detalles por parte del cliente, le pregunta a un compañero qué configuración se suele aplicar en estos casos. Este le remite a su jefa de Dpto. que le recomienda la configuración de un RAID1 gestionado con LVM, cifrando toda la información para cumplir con la legislación vigente. También le recomienda crear al menos 3 VL (hogar, raiz y swap) y una partición para el arranque.

Arquitectura de la Lección1:



RAID1 con *boot* y otro RAID1 con todo replicado. Sobre *boot a* y *boot b* haremos un RAID0 (*md0-multidevice*). Tendremos también otro RAID con el resto de información en *sda* y *sdb* y *md1* encriptado. A nivel interno, Ubuntu denomina a los md como PhysicalVolume (PV), primera capa de abstracción del sistema. Después de esto se crea el VolumeGroup (VG), que está formado por varios PhysicalVolume, esto da la facilidad de que si tenemos varios grupos de volúmenes podemos incidir internamente sobre varios discos duros. A partir de esta segunda capa de abstracción, se encuentra ya los LogicalVolume(LV), donde montaremos el */home*, la raíz del sistema */*, */var*. Y si nos quedamos sin espacio en uno de estos volúmenes lógicos es tan fácil como decirle al sistema que este volumen lógico ya no va a apuntar a *VG0*, y que pasa a apuntar

a *VG1* que va a tener un disco duro más grande, y así automáticamente hacemos que */var* ahora está en el nuevo disco duro.

Y ya por encima de los LogicalVolume, ya nos quedan las aplicaciones del sistema, que no inciden en el disco duro, internamente inciden en los LV y la información va bajando en jerarquía hasta que ya el sistema organiza los discos duros.

Primera capa de abstracción: Physical Volume (PV -- md o multidevice en Ubuntu)

-*md0* con boota y bootb y *md1* con *sda* y *sdb*.

Segunda capa de abstracción: Volume Group (VG)

-*VG0* con *md0* y *md1*

Tercera capa de abstracción: Logical Volume (LV)

Preparando máquina virtual

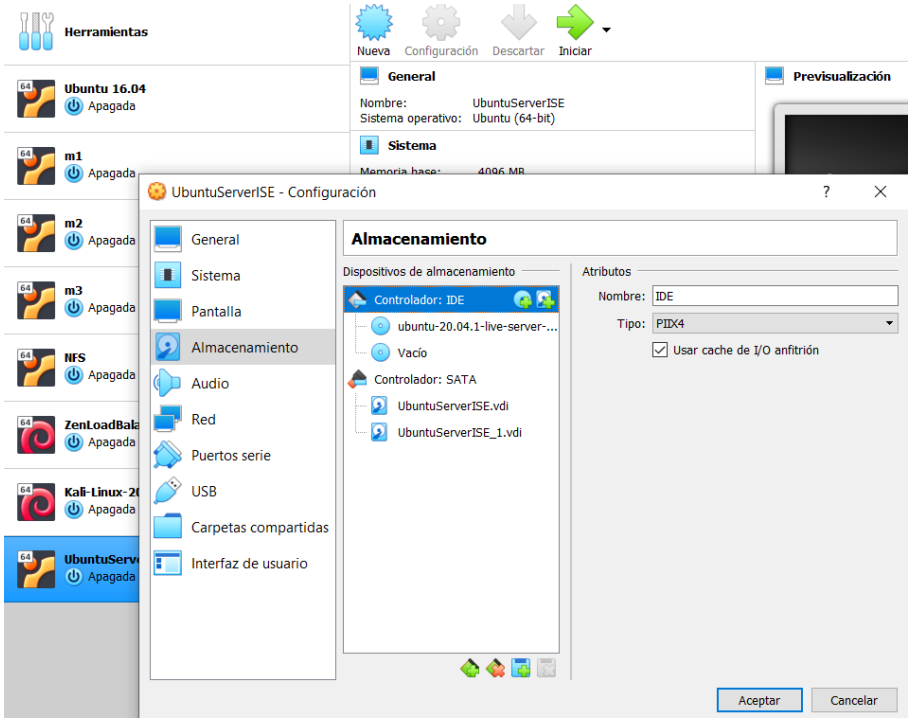
Creamos nuestra máquina virtual con UbuntuServer20.

*NOTA: En el paso siguiente de la RAM, creamos los discos duros (formato *vli*, de VirtualBox). Reservamos el espacio dinámicamente ya que es mejor ahora mismo a que te ocupe los 10GB por defecto si fuese estáticamente (que proporciona mayor velocidad al estar consecutivos los elementos en memoria)

Una vez creada creamos y asignamos los discos duros sin formatear por el momento (para los RAID). En *Configuración/Almacenamiento/*:

En *Controlador SATA*: le damos a agregar disco duro y configuramos el segundo disco duro igual que el primero.

En *Controlador IDE*: le damos a añadir unidad óptica y buscamos la imagen iso del sistema.



<u>Preparación e instalación del sistema con UbuntuServer:</u>

Iniciamos y empezamos con la instalación de UbuntuServer.

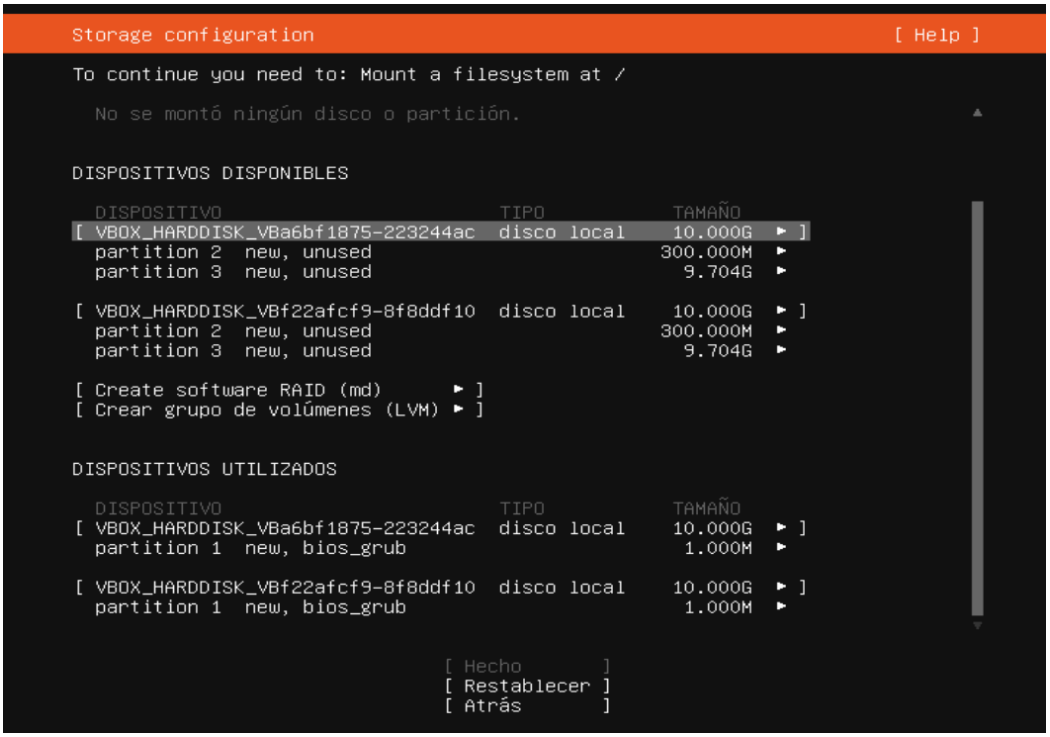
En el paso de actualización, denegamos la misma por cuestiones de estabilidad. Nunca hacer actualizaciones automáticas en servidores, siempre hacerlas con mucho cuidado e información acerca de las mismas.

Skipeamos el paso de Configuración de Conexiones de Red, dado que automáticamente VirtualBox nos "conecta" un cable de red a la máquina virtual desde el host, por lo que tenemos acceso a internet por defecto.

Skipeamos hasta llegar a *Custom Storage Layout* y empezamos la instalación customizada. Barra espaciadora para pulsar, ***tab*** para Hecho.

Configuramos la arquitectura de los discos duros del esquema anterior:

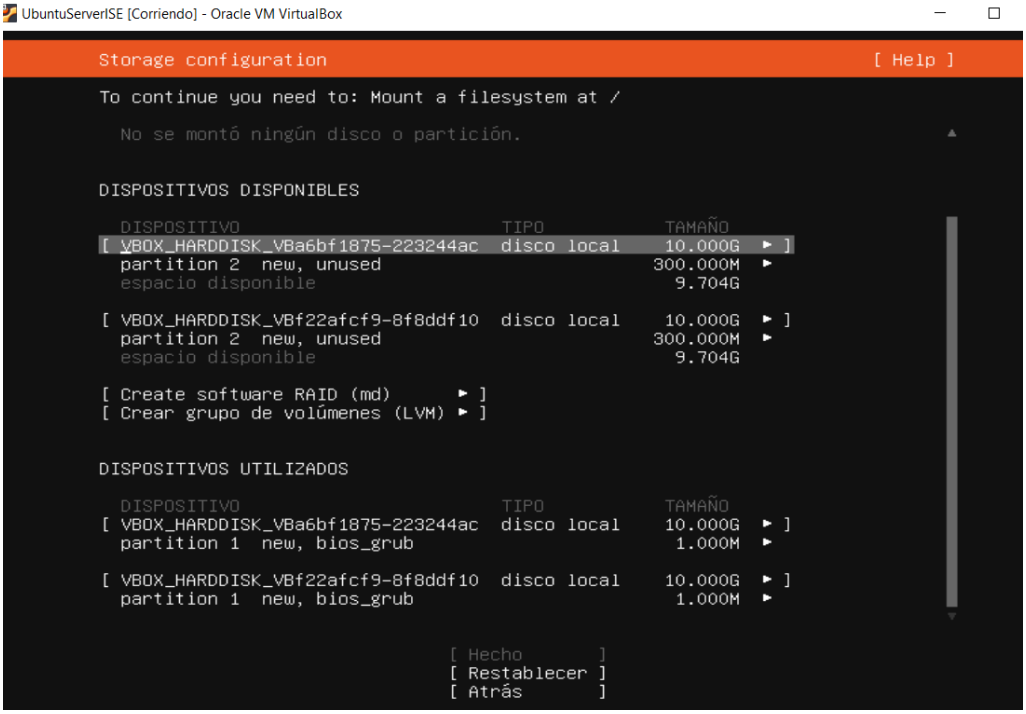
1) Necesitamos aislar */boot*: Pulsamos en el primer disco duro, le damos a *añadir GPT partition*, con 300M y sin formatear (formatearemos más adelante). Y creamos. Al hacer la primera partición, automáticamente se crea la partición *grub* para el arranque.
2) Hacemos lo mismo en el segundo disco duro, pero debemos añadir otro dispositivo de arranque *grub* debido a que es un RAID real, le damos a *Add as another boot device* para que el segundo disco duro también tenga una partición de arranque.
3) Tenemos que coger el resto de disco duro que queda y hacer una partición. Le damos al primero, *añadir partición,* como lo vamos a usar todo lo dejamos en blanco y sin formatear. Creamos.
4) Repetimos lo mismo con el segundo disco duro.
5) Una vez organizado el espacio lo primero que tenemos que crear es los RAID1. (primero discos duros, después RAID y después VG). Le damos a crear software RAID, nombre *md0* (por defecto), marcamos la partición de 300M del primer disco duro y la partición 300M del segundo disco duro para hacer el RAID1 para */boot*. Creamos *md0* con 300M.



6) Repetimos para el segundo RAID1. En *dispositivos,* marcamos la partición 3 del primer disco duro y la partición 3 del segundo disco duro. Creamos *md1* software RAID con 9`7GB.
7) Siguiente capa, crear volumen lógico VL. *Crear grupo de volúmenes LVM*. Como no hace falta añadir los dos como en el diagrama dado que no vamos a hacer nada con */boot*. Solo vamos a contener en *VG0* el RAID1 *md1*.
8) Debemos marcar la casilla de *Crear volumen cifrado*. Contraseña: practicas,ISE y creamos,.
9) Ya tenemos *VG0*, y ya no podemos hacer nada sobre md1 ya que está siendo usado por VG0, ahora ya directamente debemos incidir en VG0.
10) Primero añadimos la partición a md0 ya que tiene que tener */boot*, hemos reservado el espacio y ahora debemos crearle la partición con todo el espacio, el formato ext4 (ext4 se usa para todo en Ubuntu, *xfs* lo usa CentOS, que son tecnologías o protocolos pensados para movimientos masivos de datos en el disco duro, por lo tanto, para */boot*

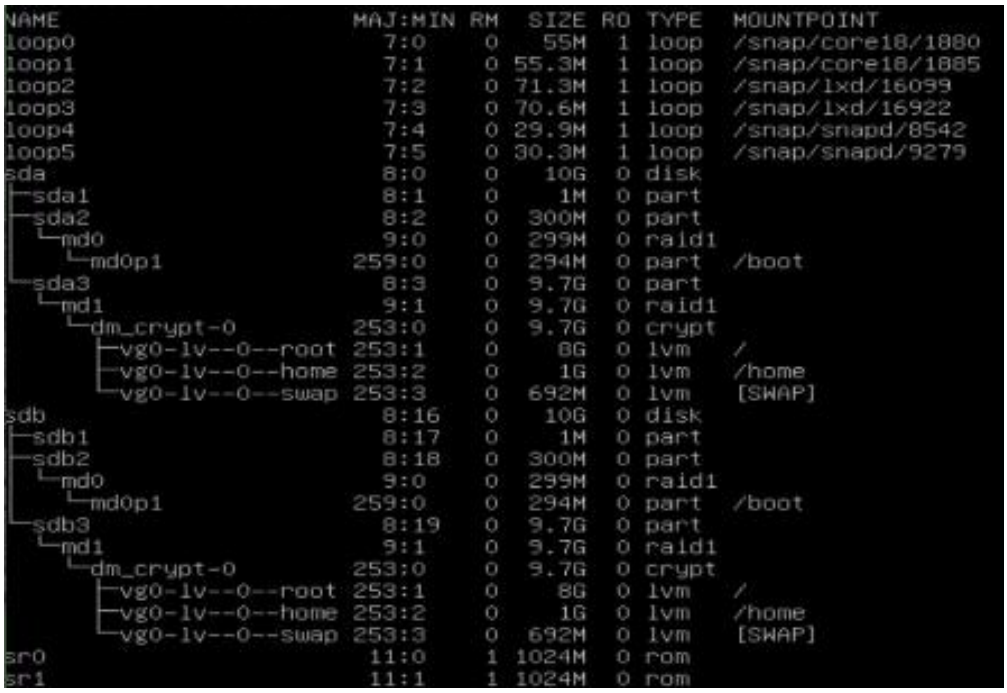
no haría falta ninguno de estos dado que solo contendrá información de inicio de sistema).

11) Le damos a *VG0* y creamos volumen lógico nuevo. La raíz, por ejemplo, ¿espacio? Para un uso de escritorio la mayoría de espacio se iría para home, pero en un servidor la información está sobre todo en */*, o */var*, etc. Por lo tanto, a raíz de daremos 8GB de momento (aunque después gracias a LVM se puede redimensionar). Formato *ext4*.
12) Una vez tenemos */boot* y */*. Creamos el siguiente volumen lógico en *VG0*, a */home* de vamos a dar 1GB porque no va a ser muy usado. Formato *ext4*.
13) Por último, la partición de */swap* para la memoria de intercambio. Formato: *swap*.



14) Ya hemos creado los dos discos duros, ya hemos creados los dos RAID, uno para */boot* y otro para todo el sistema. De ahí hemos creado un volumen lógico VL al que hemos enganchado el RAID1 *md1* (con *md0* no vamos a hacer nada). Del RAID *md0* hemos sacado */boot*, para que */boot* se monte ahí y automáticamente se replique en las particiones correspondientes de los dos discos duros. Y luego hemos ido desarrollando los volúmenes lógicos, de *VG0* hemos sacado *new LVM* de */*, de */home* y de */swap*. Como */boot* no es ningún volumen lógico aparece como *new partition of software network,* se queda como una partición.
15) Confirmamos acción destructiva y empezamos con la instalación del sistema.
16) *Usuario: peroj*
    Passwd: practicas,ISE
17) Saltamos la instalación de *OpenSSH* (ya lo haremos), igual con características. Empezamos a instalar. Cuando acabe de instalar, entre 5 y 10 minutos, hay que estar pendiente para darle a lo que sale en verde para cancelar la instalación automática y reiniciar.
18) Antes arrancar el sistema se pedirá des encriptar el disco duro con la contraseña.
19) Ya pondríamos usuario y contraseña y estaríamos dentro de nuestro sistema.

20) Para comprobar que todo se ha hecho bien: ***lsblk*** para listar como está la situación del sistema.



```
NAME                    MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
loop0                     7:0    0   55M  1 loop  /snap/core18/1880
loop1                     7:1    0 55.3M  1 loop  /snap/core18/1885
loop2                     7:2    0 71.3M  1 loop  /snap/lxd/16099
loop3                     7:3    0 70.6M  1 loop  /snap/lxd/16922
loop4                     7:4    0 29.9M  1 loop  /snap/snapd/8542
loop5                     7:5    0 30.3M  1 loop  /snap/snapd/9279
sda                       8:0    0   10G  0 disk
├─sda1                    8:1    0    1M  0 part
├─sda2                    8:2    0  300M  0 part
│ └─md0                   9:0    0  299M  0 raid1
│   └─md0p1             259:0    0  294M  0 part  /boot
└─sda3                    8:3    0  9.7G  0 part
  └─md1                   9:1    0  9.7G  0 raid1
    └─dm_crypt-0        253:0    0  9.7G  0 crypt
      ├─vg0-lv--0--root 253:1    0    8G  0 lvm   /
      ├─vg0-lv--0--home 253:2    0    1G  0 lvm   /home
      └─vg0-lv--0--swap 253:3    0  692M  0 lvm   [SWAP]
sdb                       8:16   0   10G  0 disk
├─sdb1                    8:17   0    1M  0 part
├─sdb2                    8:18   0  300M  0 part
│ └─md0                   9:0    0  299M  0 raid1
│   └─md0p1             259:0    0  294M  0 part  /boot
└─sdb3                    8:19   0  9.7G  0 part
  └─md1                   9:1    0  9.7G  0 raid1
    └─dm_crypt-0        253:0    0  9.7G  0 crypt
      ├─vg0-lv--0--root 253:1    0    8G  0 lvm   /
      ├─vg0-lv--0--home 253:2    0    1G  0 lvm   /home
      └─vg0-lv--0--swap 253:3    0  692M  0 lvm   [SWAP]
sr0                      11:0    1 1024M  0 rom
sr1                      11:1    1 1024M  0 rom
```

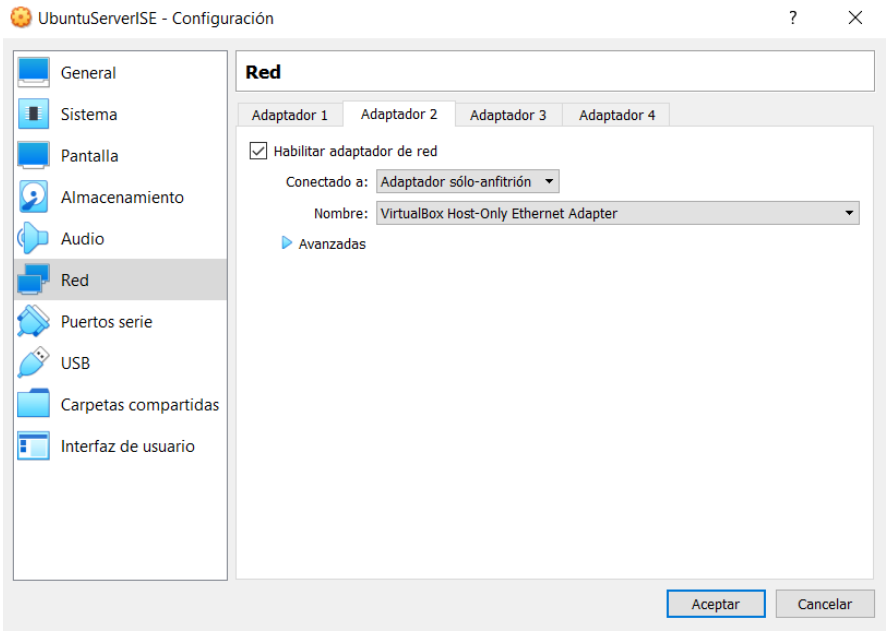
21) Y hacer sudo ***apt-get update***(o snap) e instalar el ***ifconfig*** y ***fstab***.

Configuración de red

Ahora mismo únicamente tenemos la interfaz de red que nos ha creado por defecto (enp0s3) para tener conectada nuestra máquina con internet, en modo NAT.

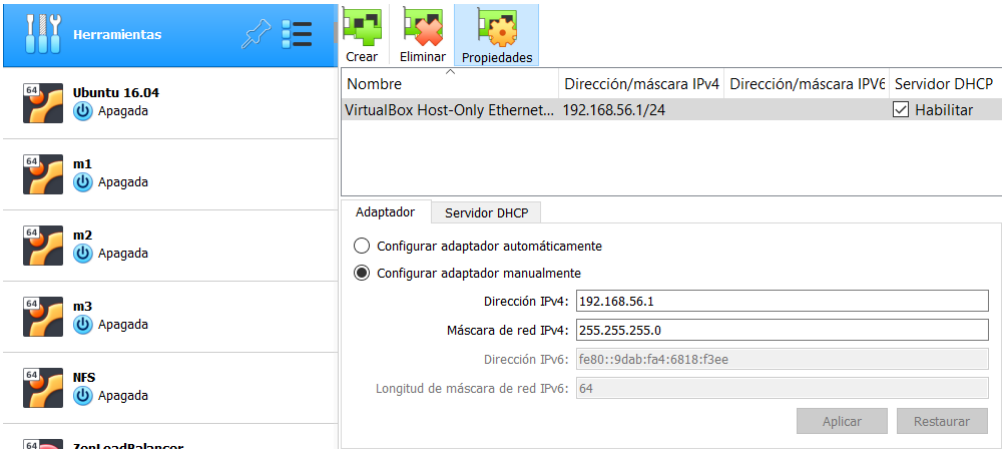
Debemos de dar de alta una interfaz de red nueva para en algún momento conectar los servidores entre sí (red local, con un cable de red conectar los equipos entre sí). Para configurar esta interfaz (enp0s8), primero debemos apagar la máquina y seguir los siguientes pasos, para dar de alta un nuevo adaptador con una tarjeta de red nueva:

1) Nos vamos a *Configuración/Red/Adaptador2/*
2) Marcamos *Habilitar adaptador de red* y lo conectamos a *Adaptador sólo-anfitrión*



Ahora debemos crear el adaptador virtual (si no lo tenemos creado ya, como en este caso), que es una interfaz que va a permitir que nuestro ordenador local o host se conecte a las máquinas virtuales y las máquinas virtuales se conecten entre sí y al host. Funciona como un *switch*, al que van conectados nuestro ordenador personal y todas las máquinas que creemos. Lo creamos de la siguiente manera:

3) Nos vamos a *Herramientas/3puntos/Red/* y pulsamos en Crear.
4) Configuramos manualmente el adaptador y queda como sigue. IP: 192.168.56.1



Ahora arrancamos la máquina, desencriptamos y entramos con nuestro usuario para ver las interfaces de red dadas de alta en el sistema y comprobar que está todo correcto.

1) Con ***lscpi | grep Ethe*** podemos ver las interfaces de red del sistema. Como vemos tenemos 2, una en el bus 3, y la otra en el bus 8.

```
peroj@peroj:~$ lspci | grep Ethe
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 02)
00:08.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 02)
peroj@peroj:~$
```

2) Debemos configurar la interfaz de red en nuestra máquina para que se conecte al adaptador enp0s8.

```
peroj@peroj:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe8e:211d  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:8e:21:1d  txqueuelen 1000  (Ethernet)
        RX packets 830  bytes 966712 (966.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 249  bytes 17647 (17.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 98  bytes 7766 (7.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 98  bytes 7766 (7.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

peroj@peroj:~$ sudo vim /etc/network/interfaces
```

3) Editamos el fichero con ***vim*** (para guardar y salir ***:wq***). La IP de nuestra máquina será estática: 192.168.56.105

```
auto enp0s8
iface enp0s8 inet static
address 192.168.56.105
~
```

4) Para aplicar el cambio tenemos dos maneras:
   i. La rápida, para levantar la interfaz de red que acabamos de crear: ***sudo ip link set enp0s8 up***
   ii. La correcta: con *netplan:*

```
peroj@peroj:~$ sudo vim /etc/netplan/00-installer-config.yaml
```

```
network:
  ethernets:
    enp0s3:
      dhcp4: true
    enp0s8:
      dhcp4: false
      addresses: [192.168.56.105/24]
  version: 2
```

Para aplicar : ***sudo netplan apply***
(no hace falta hacer el ***netplan generate*** en esta versión de Ubuntu)

Vemos con ***ifconfig*** las interfaces de red:

```
peroj@peroj:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe8e:211d  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:8e:21:1d  txqueuelen 1000  (Ethernet)
        RX packets 41  bytes 8957 (8.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 62  bytes 6956 (6.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.105  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::a00:27ff:fe60:633a  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:60:63:3a  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10  bytes 796 (796.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 88  bytes 6700 (6.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 88  bytes 6700 (6.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Comprobamos que hacemos ping a google, para comprobar que el adaptador1 (enp0s3) nos está dando conexión a internet:

```
peroj@peroj:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=19.0 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=19.3 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=20.0 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=18.1 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=116 time=18.8 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 18.053/19.031/19.991/0.639 ms
peroj@peroj:~$
```

Comprobamos si tenemos con nosotros mismos, en este caso, la IP del adaptador:

```
peroj@peroj:~$ ping 192.168.56.1
PING 192.168.56.1 (192.168.56.1) 56(84) bytes of data.
64 bytes from 192.168.56.1: icmp_seq=1 ttl=128 time=0.403 ms
64 bytes from 192.168.56.1: icmp_seq=2 ttl=128 time=0.308 ms
64 bytes from 192.168.56.1: icmp_seq=3 ttl=128 time=0.345 ms
64 bytes from 192.168.56.1: icmp_seq=4 ttl=128 time=0.341 ms
64 bytes from 192.168.56.1: icmp_seq=5 ttl=128 time=0.297 ms
^C
--- 192.168.56.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4095ms
rtt min/avg/max/mdev = 0.297/0.338/0.403/0.037 ms
```

Si hacemos ping desde PC local a la máquina:

```
C:\Users\Josele>ping 192.168.56.105

Haciendo ping a 192.168.56.105 con 32 bytes de datos:
Respuesta desde 192.168.56.105: bytes=32 tiempo=1ms TTL=64
Respuesta desde 192.168.56.105: bytes=32 tiempo<1m TTL=64
Respuesta desde 192.168.56.105: bytes=32 tiempo<1m TTL=64
Respuesta desde 192.168.56.105: bytes=32 tiempo<1m TTL=64

Estadísticas de ping para 192.168.56.105:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 0ms, Máximo = 1ms, Media = 0ms
```

LECCIÓN2: Configuración de LVM con CentOS Linux8.

2.4. Lección 2

En esta ocasión, en la empresa en la que le acaban de contratar tenían adquirido un servidor y su predecesor había realizado la instalación del S.O. CentOS, según le han comentado los compañeros, él solía hacer instalaciones por defecto y luego aplicar scripts de configuración. Sin más información, nuestro jefe nos informa que esa máquina va a alojar unos cursos con vídeos de alta calidad y relativamente largos. Por tanto, viendo la configuración del sistemas, prevemos que /var necesitará más espacio, incluso es conveniente asignarle un LV exclusivamente. Para ello, incluiremos un nuevo disco y configuraremos LVM para que /var se monte en el nuevo VL que crearemos para él.

En esta lección, haremos algo parecido con CentOS. Nos centraremos únicamente en el LVM con CentOS y más adelante montaremos los RAID. Haremos instalación por defecto.

Tendremos que hacer uso de LVM a posteriori, primero instalamos el sistema y luego lo configuraremos (no como en Ubuntu que estructuramos todo correctamente a priori) para ampliar */var* como nos dice el caso práctico en este caso.

En el disco duro a, nos quedamos sin espacio, y como no se ha previsto, debemos configurarlo sobre la marcha. Crearemos un disco duro b y cambiaremos el puntero de */var* para que apunte a b y no a a.

Por tanto, en esta parte de la lección haremos:

- Instalación por defecto de CentOS
- Configurar LV (logical volume) solo para */var* en un disco duro nuevo para ampliar el */var* ya que nos hemos quedado sin espacio.

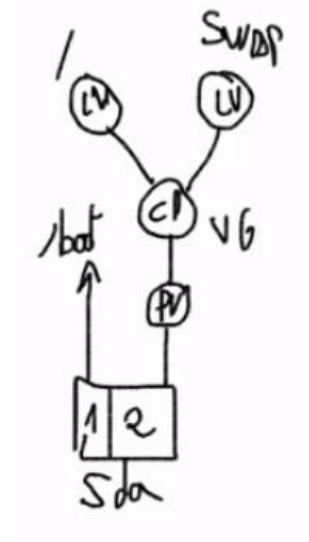
<u>Arquitectura de la Lección2:</u>

Tenemos un disco duro, que es *sda*. CentOS por defecto hace dos particiones (*sda1* y *sda2*). En *sda1* directamente se crea la partición para */boot*, sin volumen lógico (cuidado, que le asigna 1GB y es demasiado, habrá que redimensionarlo). En *sda2*, que es el resto de datos del sistema si crea un Physical Volume (primera capa de abstracción del sistema operativo de un dispositivo físico). En la Lección1 no hacíamos ningún PV de los discos duros porque teníamos un RAID y este servía de abstracción de los dos discos duros).

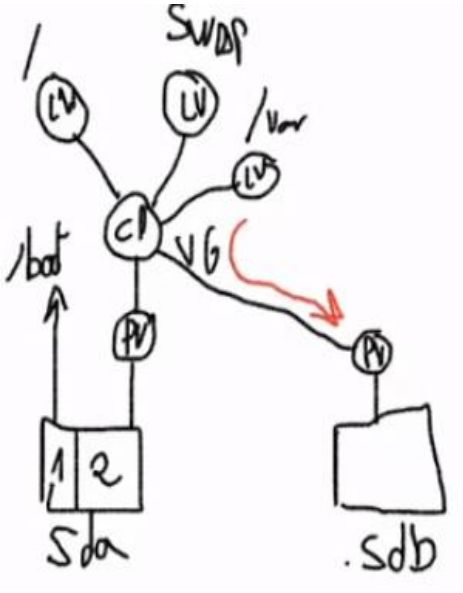
En ese PV, también CentOS por defecto crea un Volume Group o *cl* con la nomenclatura de CentOS (en Ubuntu era *vg*). Este grupo de volumenes ahora mismo solo tiene el PV de *sda2*. Por encima, CentOS nos da dos Logical Volume ya configurados, uno para */swap* y otro para el resto del sistema */*.

Se nos pide meter un disco duro nuevo *sdb* y que creemos un nuevo volumen lógico para */var.* Es decir, meter sin partición *sdb,* hacer PV y enganchar al grupo de volúmenes ya creado durante la instalación, para interconectar *sda* y *sdb*. Ahora debemos crear otro volumen lógico en el grupo de volumenes *cl* para */var*.

Una vez que esté */var* creado, le vamos a decir internamente que el volumen lógico de /var va a ir montado en *sdb* y no en *sda* (que suponemos completo). Así quedaría redimensionado */var* sobre la marcha.



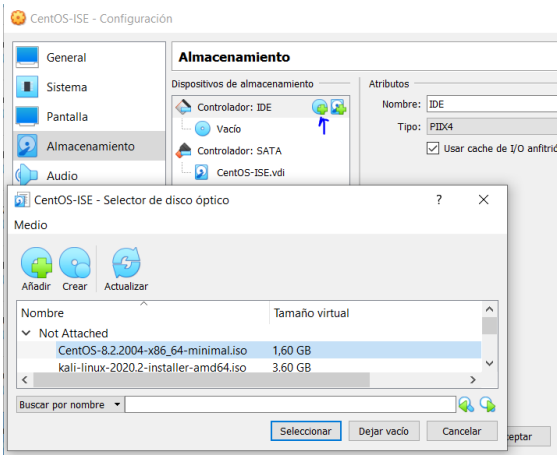
A) Arquitectura instalación por defecto de CentOS.



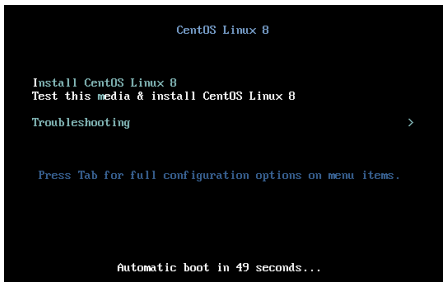
B) Arquitectura final de la Lección2.

Preparación e instalación del sistema con CentOS:

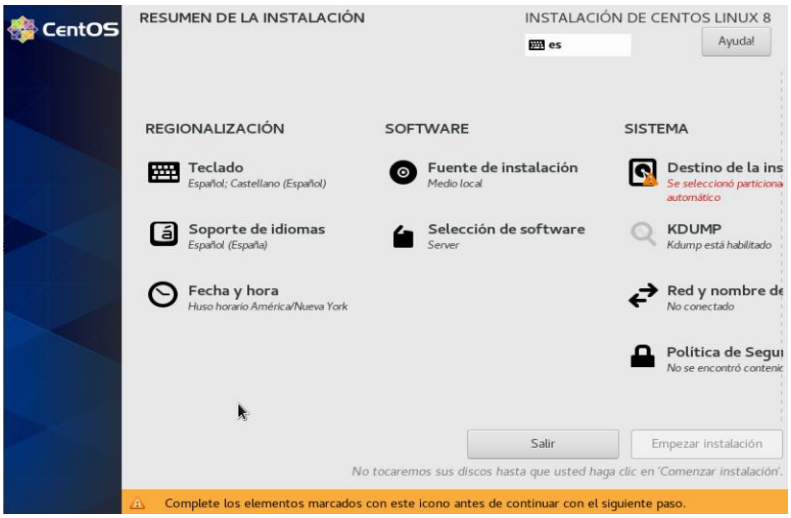
1) Creamos una nueva máquina en VirtualBox como siempre (es decir, 4GB RAM y 8GB disco duro virtual *vdi* reservado dinámicamente).
2) Seguidamente pasamos a cargar la iso en *Configuración/Almacenamiento/Controlador:IDE/AgregarUnidadOptica*



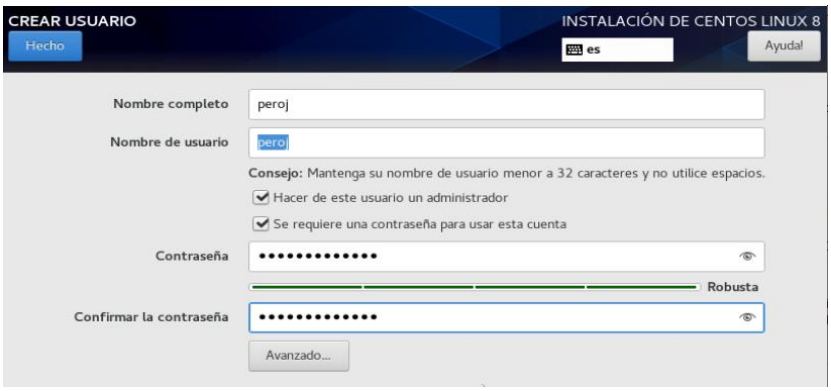
3) Iniciamos la máquina y seleccionamos Install CentOS y veremos que la instalación por defecto es mucho más gráfica que la de UbuntuServer (para salir del modo captura de ratón de VirtualBox pulsamos *ctrl* derecho, tecla de anfitrión).



4) Después de seleccionar idioma (nos movemos por los menús tabulando), nos saldrán las diferentes opciones del sistema, debemos de tabular hasta *Destino de la instalación* que tiene warning por seguridad (para asegurarse de que queremos instalarlo en ese lugar). Una vez entremos y salgamos se quitará el warning y continuamos.



5) Mientras avanza la instalación, ponemos la Contraseña de root: practicas,ISE . y vamos creando nuestro usuario (peroj) con la misma contraseña. Es importante marcar la casilla de *Hacer de este usuario un administrador* dado que sino el sistema no te incluirá en el fichero *sudoers* y no te permitirá hacer *sudo* en ningún momento (solo podrás hacerlo desde la cuenta de usuario root).



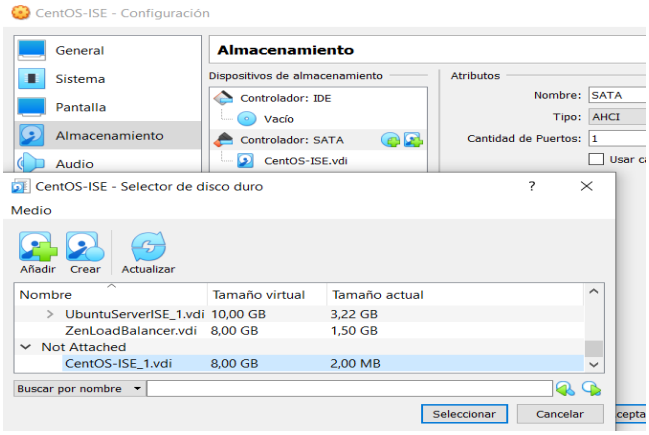
6) Cuando termine la instalación debemos reiniciar, pero previamente debemos desmontar la unidad óptica con la iso desde *Configuración/Almacenamiento/...*(sacar el disco), ya que será lo que abra por defecto al reiniciar (a diferencia de Ubuntu). Ahora si iniciamos, y en *grub* elegimos la primera opción. Carga y ya tenemos el login.
7) Entramos con nuestro usuario y comprobamos el estado del sistema con ***lsblk***: vemos que tenemos el esquema de la arquitectura A.

```
CentOS Linux 8 (Core)
Kernel 4.18.0-193.el8.x86_64 on an x86_64

Activate the web console with: systemctl enable --

localhost login: [   25.297702] snd_intel8x0 0000:
peroj
Password:
[peroj@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 1024M  0 rom
[peroj@localhost ~]$ _
```

8) Nos damos cuenta que la partición de */boot* es demasiado grande, dado que la capacidad necesaria para los archivos de arranque del sistema (core del SO) es mínima (300MB es suficiente).
9) Para incluir el segundo disco duro *sdb* debemos apagar la máquina y *Configuración/Almacenamiento/Controlador:SATA/Agregar disco duro.* Y por defecto.



10) Iniciamos y comprobamos:

```
[peroj@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
sr0          11:0    1 1024M  0 rom
```

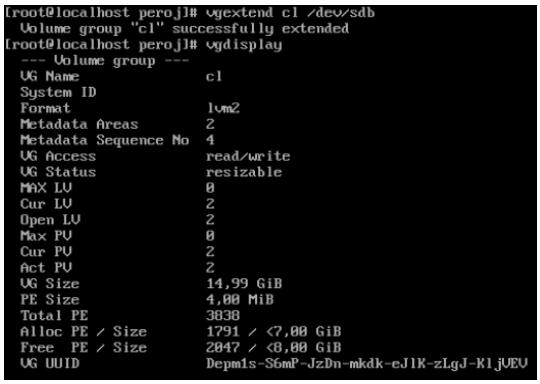
11) A partir de aquí vamos a crear el Physical Volumen y le vamos a decir que está dentro del grupo de volúmenes *cl* (grupo creado por defecto durante la instalación). Y después vamos a montar un Logical Volume para */var* y vamos a montar /var en el segundo disco duro. Lo podemos hacer desde la consola de la herramienta ***lvm*** (*man lvm*[1]) o independientemente desde nuestro terminal (***sudo su***).
12) Para crear el Physical Volume del *sdb* usamos: ***pvcreate /dev/sdb*** (en */dev* están todos los dispositivos). Una vez que lo tenemos creado, lo comprobamos con ***pvdisplay***. No tiene memoria debido a que todavía no tiene formato.

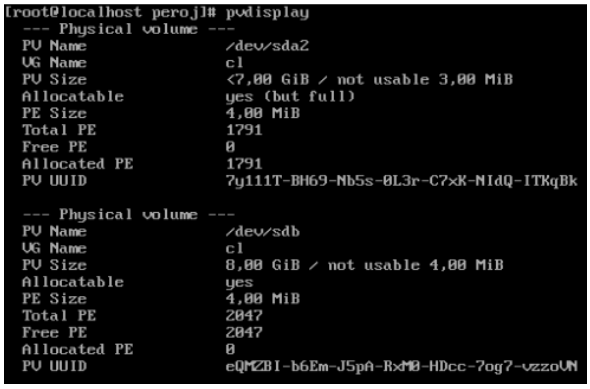
```
[peroj@localhost ~]$ sudo su
[sudo] password for peroj:
[root@localhost peroj]# pvcreate /dev/sdb
  Physical volume "/dev/sdb" successfully created.
[root@localhost peroj]# pvdisplay
  --- Physical volume ---
  PV Name               /dev/sda2
  VG Name               cl
  PV Size               <7,00 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               7y111T-BH69-Nb5s-0L3r-C7xK-NIdQ-ITKqBk

  "/dev/sdb" is a new physical volume of "8,00 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdb
  VG Name
  PV Size               8,00 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               eQMZBI-b6Em-J5pA-RxM0-HDcc-7oy7-vzzoVM
```

[1] para avanzar en el *man* hasta la última página donde podemos encontrar ejemplos pulsamos ***máyus+g*** y ***q*** para salir.
Si tabulamos en la consola de ***lvm*** nos aparecen todas sus herramientas disponibles.

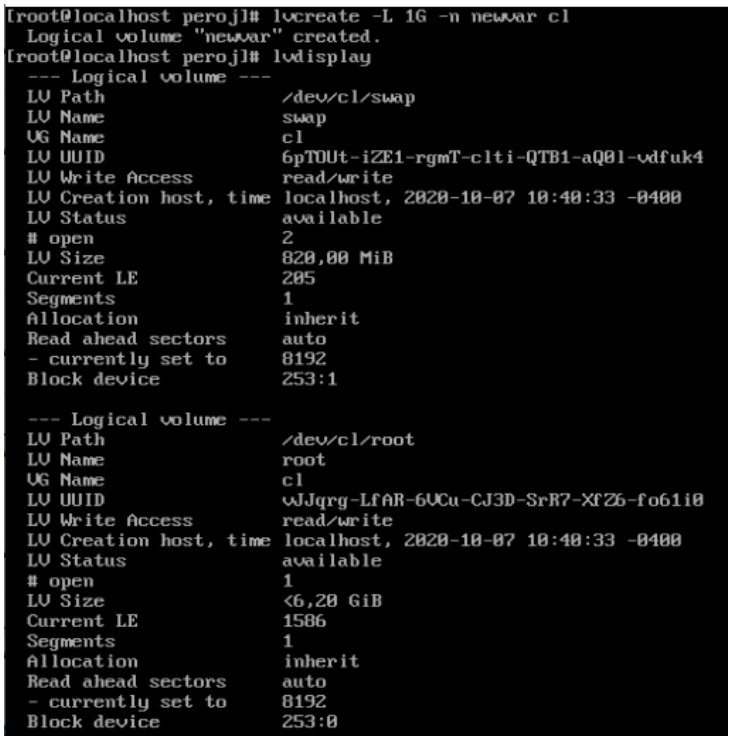
13) Ahora enganchamos este volumen físico al grupo de volúmenes *cl* con:
vgextend cl /dev/sdb , para comprobarlo ***vgdisplay*** aunque habrá que deducir cuál es por los datos de memoria. Con ***pvdisplay*** podemos ver mejor que en el segundo atributo de cada PV aparece el VG al que está enganchado, en este caso *cl*. Continuamos.





Con ***vgdisplay*** Con ***pvdisplay*** (recomendado)

14) Por lo tanto, siguiendo el orden ascendente (como siempre) en el esquema de la arquitectura B, empezamos con los discos duros, hacemos la abstracción a los volúmenes físicos, hacemos la abstracción a los grupos de volúmenes y hacemos la abstracción a los volúmenes lógicos.
Entonces ahora tendríamos que crear el volumen lógico para */var*, con ***lvcreate***. Con ***-L*** (de large) asignamos la capacidad (le damos 1GB de prueba). Con ***-n*** le damos el nombre *newvar.* Y por último el grupo de volúmenes al que va a pertenecer, *cl.* (tamaño, nombre, grupo al que va*).*



```
--- Logical volume ---
LV Path                /dev/cl/newvar
LV Name                newvar
VG Name                cl
LV UUID                4vF0LV-ZegP-eHS0-OrUu-KYBh-exoc-sV3tc4
LV Write Access        read/write
LV Creation host, time localhost.localdomain, 2020-10-07 12:06:30 -0400
LV Status              available
# open                 0
LV Size                1,00 GiB
Current LE             256
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     8192
Block device           253:2
```
[2]

15) Hasta aquí ya tendríamos los niveles de abstracción creados, pero en el disco duro *sdb* el volumen lógico no tiene ningún tipo de formato. Por lo tanto, le damos formato al volumen lógico *newvar* con ***mkfs*****,** para construir un sistema de ficheros en Linux. Lo podemos hacer de dos maneras:
    a. ***mkfs -t ext4 /dev/cl/newvar*** Forma típica
    b. ***mkfs -t ext4 /dev/mapper/cl-newvar*** El modulo mapper hace un mapeado de todos los dispositivos que hay en el sistema.

    Con ***-t*** indicamos el formato, tanto *ext4* como *xfs* son recomendables para mover grandes cantidades de datos.

    Comprobamos con ***lsblk***.

```
[root@localhost perojl]# mkfs -t ext4 /dev/mapper/cl-newvar
mke2fs 1.45.4 (23-Sep-2019)
Se está creando un sistema de ficheros con 262144 bloques de 4k y 65536 nodos-i
UUID del sistema de ficheros: e822e0d0-c653-4777-9a94-7245eedbdf86
Respaldos del superbloque guardados en los bloques:
        32768, 98304, 163840, 229376

Reservando las tablas de grupo: hecho
Escribiendo las tablas de nodos-i: hecho
Creando el fichero de transacciones (8192 bloques): hecho
Escribiendo superbloques y la información contable del sistema de ficheros: hecho

[root@localhost perojl]# 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
└─cl-newvar 253:2    0    1G  0 lvm
sr0          11:0    1 1024M  0 rom
```

Podemos ver que en *MOUNTPOINT* no pone nada dado que no lo hemos montado en ningún sitio, únicamente lo hemos creado.

[2] para movernos en la terminal arriba y abajo: ***mayus+avpag*** ó ***mayus+repag***

16) Para decirle que el punto de montaje es */var* debemos seguir los siguientes pasos:
    a. Copiar */var* a */newvar* (previamente montado */newvar*)

Creamos un punto de montaje porque para hacer la copia /newvar debe estar montado en el sistema (dado que ahora mismo no se puede acceder por ningún lado a su información). Creamos carpeta donde montaremos /newvar en */mnt* con ***mount*** (que monta un sistema de ficheros en un directorio concreto).

```
[root@localhost perojl]# mkdir mkdir /mnt/newvar
[root@localhost perojl]# ls /mnt/
newvar
[root@localhost perojl]# mount /dev/cl/newvar /mnt/newvar
[ 8168.889835] EXT4-fs (dm-2): mounted filesystem with ordered data mode. Opts: (null)
[root@localhost perojl]#
```

Ahora copiamos */var* a */newvar*. Para hacer la copia recursiva de sus directorios ***-r***. No basta sólo con eso, debemos tener mucho cuidado con el Contexto de cada archivo (para verlos ***ls -Z)*** y si se copia solo de forma recursiva no se copiarán los contextos y no serán iguales los *strings*. Esto se debe a un módulo de Linux, *SELinux (Security Echanced Linux)* que incluye muchas cosas de seguridad, entre ellas, los Contextos, que indica si un archivo ha sido "tocado", y si detecta que la carpeta no tiene su contexto de seguridad, *SELinux* saltará. Para copiar también los contextos, usaremos ***cp -a*** que tiene en cuenta directorios, recursivo (***-dR***) y el ***--preserve=ALL,*** gracias a este último este *cp* te mantiene la carpeta tal cual, con sus contextos y todo.

Tenemos el problema añadido de que, al ser un servidor, si mientras estamos copiando un usuario introduce algo en el */var*, los dos no quedarían iguales al final, no sería una copia atómica. Por lo que deberemos cambiar el modo de usuario a modo mantenimiento (nivel de ejecución 1) para echar a todos los usuarios del sistema, hacer la copia de seguridad y salir del modo mantenimiento.

Para esto usaremos ***systemctl*** que controla el sistema y los servicios del sistema. Para saber en qué modo de ejecución estamos usamos ***systemctl status*** (running). Para entrar en modo mantenimiento usaremos ***systemctl isolate rescue***[3]. Metemos contraseña y comprobamos que estamos en modo mantenimiento y hacemos la copia ***cp -a /var/. /mnt/newvar*** (con el **.** para copiar todo el contenido y sin la barra al final porque si no, crearía una carpeta dentro de */newvar*). Comprobamos que todos los contextos son iguales y se ha copiado todo correctamente y salimos del modo mantenimiento con ***systemctl default***. Para proseguir entramos con nuestro usuario y comprobamos (***systemctl status)***.

[3] Bug en la versión nueva CentOS8 al ejecutar este comando dado que parece que lo va a hacer pero nos saca a la pantalla de login. En este momento ingresamos como root, dado que sino no podremos ejecutar la orden correctamente.

```
[root@localhost ~]# mount /dev/cl/newvar /mnt/newvar
mount: /mnt/newvar: /dev/mapper/cl-newvar ya está montado en /mnt/newvar.
[root@localhost ~]# cp -a /var/. /mnt/newvar
[root@localhost ~]# ls -Z /var
     system_u:object_r:acct_data_t:s0 account             system_u:object_r:var_t:s0 local
           system_u:object_r:var_t:s0 adm               system_u:object_r:var_lock_t:s0 lock
           system_u:object_r:var_t:s0 cache             system_u:object_r:var_log_t:s0 log
   system_u:object_r:kdump_crash_t:s0 crash           system_u:object_r:mail_spool_t:s0 mail
     system_u:object_r:system_db_t:s0 db                    system_u:object_r:var_t:s0 nis
           system_u:object_r:var_t:s0 empty                 system_u:object_r:var_t:s0 opt
system_u:object_r:public_content_t:s0 ftp                   system_u:object_r:var_t:s0 preserve
    system_u:object_r:games_data_t:s0 games               system_u:object_r:var_run_t:s0 run
           system_u:object_r:var_t:s0 gopher            system_u:object_r:var_spool_t:s0 spool
           system_u:object_r:var_t:s0 kerberos                system_u:object_r:tmp_t:s0 tmp
       system_u:object_r:var_lib_t:s0 lib                 system_u:object_r:var_yp_t:s0 yp
[root@localhost ~]# ls -Z /newvar
ls: no se puede acceder a '/newvar': No such file or directory
[root@localhost ~]# ls -Z /mnt/newvar
     system_u:object_r:acct_data_t:s0 account           system_u:object_r:var_lock_t:s0 lock
           system_u:object_r:var_t:s0 adm               system_u:object_r:var_log_t:s0 log
           system_u:object_r:var_t:s0 cache           system_u:object_r:unlabeled_t:s0 lost+found
   system_u:object_r:kdump_crash_t:s0 crash           system_u:object_r:mail_spool_t:s0 mail
     system_u:object_r:system_db_t:s0 db                    system_u:object_r:var_t:s0 nis
           system_u:object_r:var_t:s0 empty                 system_u:object_r:var_t:s0 opt
system_u:object_r:public_content_t:s0 ftp                   system_u:object_r:var_t:s0 preserve
    system_u:object_r:games_data_t:s0 games               system_u:object_r:var_run_t:s0 run
           system_u:object_r:var_t:s0 gopher            system_u:object_r:var_spool_t:s0 spool
           system_u:object_r:var_t:s0 kerberos                system_u:object_r:tmp_t:s0 tmp
       system_u:object_r:var_lib_t:s0 lib                 system_u:object_r:var_yp_t:s0 yp
           system_u:object_r:var_t:s0 local
```

b. Montar, comprobar y desmontar */newvar*, dado que luego lo montaremos y desmontaremos de forma automatizada desde el fichero *fstab*.

c. Modificar fichero *fstab* --> En este fichero pondremos los puntos de montaje que queremos (que ya estarán */boot* y */swap* creados automáticamente por CentOS), añadimos /var para que cuando el sistema arranque lea el fichero y lo monte automáticamente al arrancar el sistema. (modificamos el fichero con ***vi /etc/fstab***).

```
#
# /etc/fstab
# Created by anaconda on Wed Oct  7 10:40:37 2020
#
# 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=b59f3979-a6cc-4ab5-9626-4a8062a80139 /boot                   ext4    defaults        1 2
/dev/mapper/cl-swap     swap                    swap    defaults        0 0


/dev/mapper/cl-newvar   /var                    ext4    defaults        0 0
~
```

Los dos últimos bits de cada fila establecen nivel de prioridad al hacer backups.

Para comprobar que se monta bien, primero vamos a desmontarlo con ***umount*** (si diese problemas de que está ocupado usamos la opción ***-l*** que irá desocupando el dispositivo poco a poco hasta desmontarlo).

Ahora lo montamos con ***mount -a*** (que monta leyendo desde el fichero *fstab*).

```
[root@localhost peroj]# umount /mnt/newvar
umount: /mnt/newvar: no montado.
[root@localhost peroj]# umount -l /mnt/newvar
umount: /mnt/newvar: no montado.
[root@localhost peroj]# mount -a
[75308.351487] EXT4-fs (dm-2): mounted filesystem with ordered data mode. Opts: (null)
```

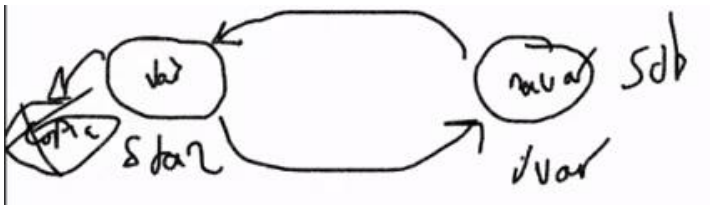
Comprobamos con ***mount*** si lo ha montado (en la última línea) y con ***lsblk*** (y vemos que el punto de montaje está bien con /var).

```
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,se
,gid=1000)
/dev/mapper/cl-newvar on /var type ext4 (rw,relatime,seclabel)
[root@localhost peroj]# 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
└─cl-newvar 253:2    0    1G  0 lvm  /var
sr0           11:0    1 1024M  0 rom
[root@localhost peroj]#
```

d. Liberar espacio del antiguo */var*

   i. Hacer BackUp (momentáneo) de */var* por si hubiese algún problema posterior a la eliminación.

Ahora ya tenemos montado */var* y no podemos acceder al antiguo */var* ya que está en el disco duro a (*sda2*) y ese */var* esta inaccesible porque al /var ahora se accede desde *sdb*. Entonces si queremos hacer una copia tenemos que invertir el proceso, es decir, quitar otra vez el ***mount*** anteriror para que se monte automáticamente el */var* antiguo, copiamos el */var* antiguo y volvemos a hacer el ***mount*** anterior. Proceso:




```
[root@localhost peroj]# umount /dev/mapper/cl-newvar
[root@localhost peroj]# 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
└─cl-newvar 253:2    0    1G  0 lvm
sr0           11:0    1 1024M  0 rom
[root@localhost peroj]# mv /var /var_OLD
[root@localhost peroj]# ls -Z /
      system_u:object_r:bin_t:s0 bin          system_u:object_r:proc_t:s0 proc
     system_u:object_r:boot_t:s0 boot   system_u:object_r:admin_home_t:s0 root
   system_u:object_r:device_t:s0 dev       system_u:object_r:var_run_t:s0 run
      system_u:object_r:etc_t:s0 etc           system_u:object_r:bin_t:s0 sbin
 system_u:object_r:home_root_t:s0 home         system_u:object_r:var_t:s0 srv
      system_u:object_r:lib_t:s0 lib        system_u:object_r:sysfs_t:s0 sys
      system_u:object_r:lib_t:s0 lib64        system_u:object_r:tmp_t:s0 tmp
      system_u:object_r:mnt_t:s0 media        system_u:object_r:usr_t:s0 usr
      system_u:object_r:mnt_t:s0 mnt          system_u:object_r:var_t:s0 var_OLD
      system_u:object_r:usr_t:s0 opt
[root@localhost peroj]# mount -a
mount: /var: el punto de montaje no existe.
```

Como */var* no existe, lo creamos a mano con ***mkdir /var,*** pero si comprobamos los contextos no serán iguales por lo que tendremos que usar ***restorecon*** (para restaurarlos).

```
[root@localhost peroj]# mkdir /var
[root@localhost peroj]# ls -Z /
        system_u:object_r:bin_t:s0 bin           system_u:object_r:proc_t:s0 proc
       system_u:object_r:boot_t:s0 boot     system_u:object_r:admin_home_t:s0 root
     system_u:object_r:device_t:s0 dev        system_u:object_r:var_run_t:s0 run
        system_u:object_r:etc_t:s0 etc           system_u:object_r:bin_t:s0 sbin
  system_u:object_r:home_root_t:s0 home          system_u:object_r:var_t:s0 srv
        system_u:object_r:lib_t:s0 lib         system_u:object_r:sysfs_t:s0 sys
        system_u:object_r:lib_t:s0 lib64         system_u:object_r:tmp_t:s0 tmp
        system_u:object_r:mnt_t:s0 media         system_u:object_r:usr_t:s0 usr
        system_u:object_r:mnt_t:s0 mnt     unconfined_u:object_r:default_t:s0 var
        system_u:object_r:usr_t:s0 opt           system_u:object_r:var_t:s0 var_OLD
[root@localhost peroj]# restorecon /var
[root@localhost peroj]# ls -Z /
       system_u:object_r:bin_t:s0 bin           system_u:object_r:proc_t:s0 proc
      system_u:object_r:boot_t:s0 boot     system_u:object_r:admin_home_t:s0 root
    system_u:object_r:device_t:s0 dev        system_u:object_r:var_run_t:s0 run
       system_u:object_r:etc_t:s0 etc           system_u:object_r:bin_t:s0 sbin
 system_u:object_r:home_root_t:s0 home          system_u:object_r:var_t:s0 srv
       system_u:object_r:lib_t:s0 lib         system_u:object_r:sysfs_t:s0 sys
       system_u:object_r:lib_t:s0 lib64         system_u:object_r:tmp_t:s0 tmp
       system_u:object_r:mnt_t:s0 media         system_u:object_r:usr_t:s0 usr
       system_u:object_r:mnt_t:s0 mnt       unconfined_u:object_r:var_t:s0 var
       system_u:object_r:usr_t:s0 opt           system_u:object_r:var_t:s0 var_OLD
```

El *unconfined_u* es normal después de la restauración del contexto. Realmente ahora cuando montemos el */newvar*, como va a coger el */var* que tiene en *sdb*, automáticamente el contexto va a volver al que está en *sdb*, que si es exactamente igual.

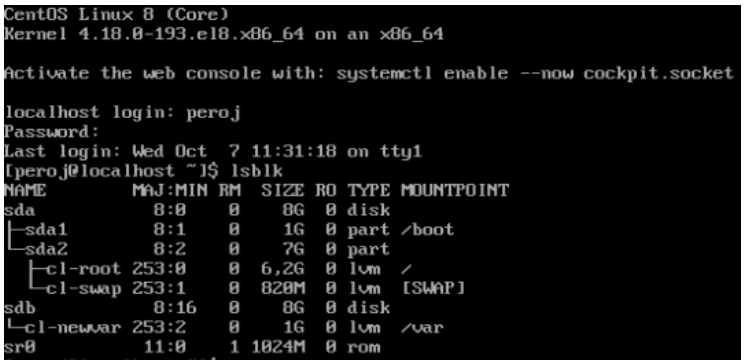
Una vez hecho esto, hacemos el ***mount -a*** y comprobamos (con ***mount*** y ***lsblk***).

```
[root@localhost peroj]# mount -a
[77426.172709] EXT4-fs (dm-2): mounted filesystem with ordered data mode. Opts: (null)
[root@localhost peroj]# 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
└─cl-newvar 253:2    0    1G  0 lvm  /var
sr0          11:0    1 1024M  0 rom
```

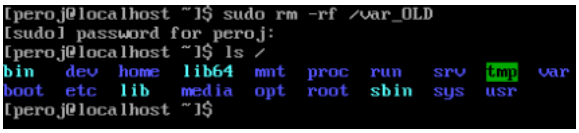
Y comprobamos los contextos (el *unconfined* ya no está) y tenemos accesible el */var_OLD* (comprobamos los contextos entre sí)*:*

```
[root@localhost peroj]# ls -Z /
       system_u:object_r:bin_t:s0 bin           system_u:object_r:proc_t:s0 proc
      system_u:object_r:boot_t:s0 boot     system_u:object_r:admin_home_t:s0 root
    system_u:object_r:device_t:s0 dev        system_u:object_r:var_run_t:s0 run
       system_u:object_r:etc_t:s0 etc           system_u:object_r:bin_t:s0 sbin
 system_u:object_r:home_root_t:s0 home          system_u:object_r:var_t:s0 srv
       system_u:object_r:lib_t:s0 lib         system_u:object_r:sysfs_t:s0 sys
       system_u:object_r:lib_t:s0 lib64         system_u:object_r:tmp_t:s0 tmp
       system_u:object_r:mnt_t:s0 media         system_u:object_r:usr_t:s0 usr
       system_u:object_r:mnt_t:s0 mnt           system_u:object_r:var_t:s0 var
       system_u:object_r:usr_t:s0 opt           system_u:object_r:var_t:s0 var_OLD
[root@localhost peroj]# ls -Z /var
    system_u:object_r:acct_data_t:s0 account         system_u:object_r:var_lock_t:s0 lock
           system_u:object_r:var_t:s0 adm             system_u:object_r:var_log_t:s0 log
           system_u:object_r:var_t:s0 cache         system_u:object_r:unlabeled_t:s0 lost+found
  system_u:object_r:kdump_crash_t:s0 crash        system_u:object_r:mail_spool_t:s0 mail
    system_u:object_r:system_db_t:s0 db                system_u:object_r:var_t:s0 nis
           system_u:object_r:var_t:s0 empty             system_u:object_r:var_t:s0 opt
system_u:object_r:public_content_t:s0 ftp               system_u:object_r:var_t:s0 preserve
   system_u:object_r:games_data_t:s0 games           system_u:object_r:var_run_t:s0 run
           system_u:object_r:var_t:s0 gopher        system_u:object_r:var_spool_t:s0 spool
           system_u:object_r:var_t:s0 kerberos            system_u:object_r:tmp_t:s0 tmp
      system_u:object_r:var_lib_t:s0 lib            system_u:object_r:var_yp_t:s0 yp
           system_u:object_r:var_t:s0 local
[root@localhost peroj]# ls -Z /var_OLD
    system_u:object_r:acct_data_t:s0 account             system_u:object_r:var_t:s0 local
           system_u:object_r:var_t:s0 adm           system_u:object_r:var_lock_t:s0 lock
           system_u:object_r:var_t:s0 cache           system_u:object_r:var_log_t:s0 log
  system_u:object_r:kdump_crash_t:s0 crash        system_u:object_r:mail_spool_t:s0 mail
    system_u:object_r:system_db_t:s0 db                system_u:object_r:var_t:s0 nis
           system_u:object_r:var_t:s0 empty             system_u:object_r:var_t:s0 opt
system_u:object_r:public_content_t:s0 ftp               system_u:object_r:var_t:s0 preserve
   system_u:object_r:games_data_t:s0 games           system_u:object_r:var_run_t:s0 run
           system_u:object_r:var_t:s0 gopher        system_u:object_r:var_spool_t:s0 spool
           system_u:object_r:var_t:s0 kerberos            system_u:object_r:tmp_t:s0 tmp
      system_u:object_r:var_lib_t:s0 lib            system_u:object_r:var_yp_t:s0 yp
[root@localhost peroj]#
```

Hacemos ***reboot*** y comprobamos (con ***mount y lsblk***) que cuando el sistema arranca lo hace todo desde cero antes de liberar el espacio.



ii. Liberar *var_OLD* cuando estemos completamente seguros de que el sistema es estable. Para forzarlo con: ***rm -rf /var_OLD***



LECCIÓN3: Configuración de RAIDs en CentOS Linux8 y cifrado.

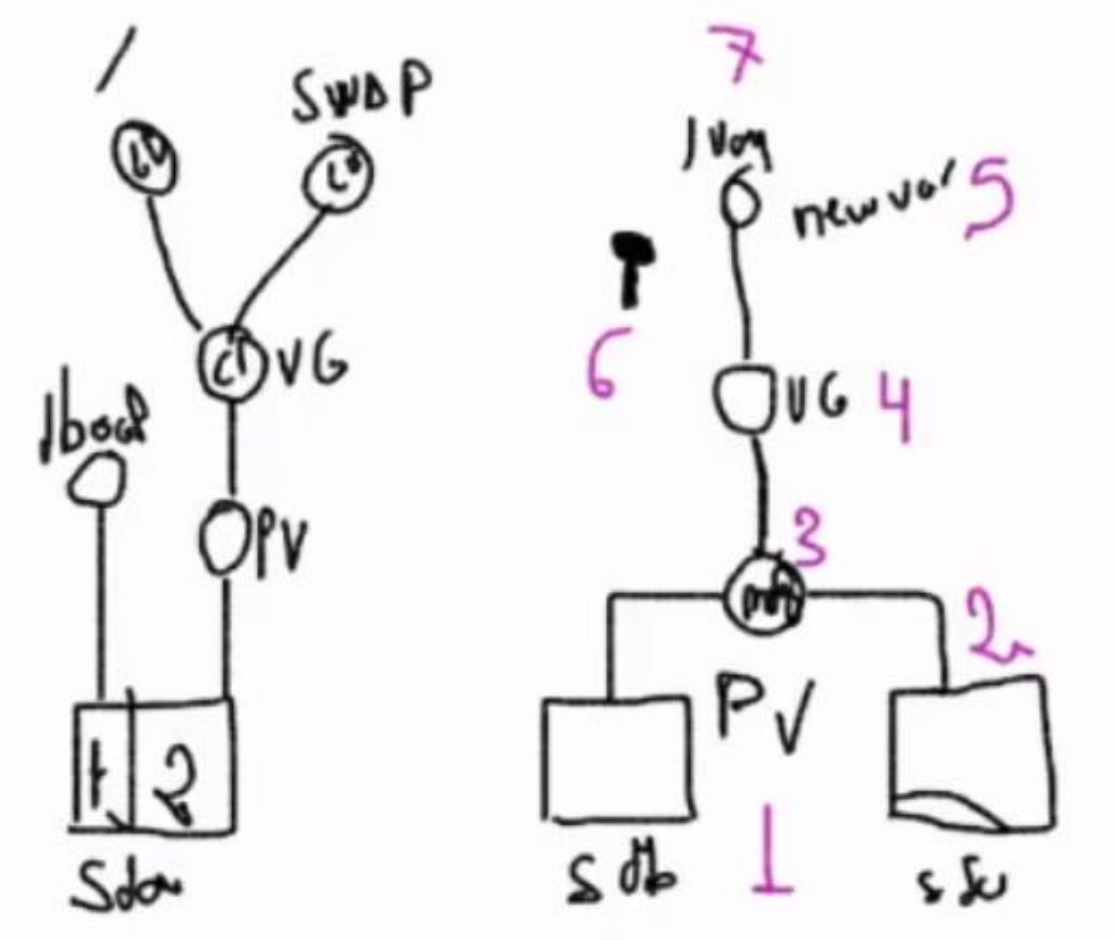
2.5. Lección 3

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.

Breve descripción de pasos a seguir (después de instalar CentOS por defecto):

1. Insertar los discos duros (*sdb* y *sdc*) donde implementaremos el RAID.
2. Crear el RAID1 con la herramienta *mdadm*
3. Crear PV *md0*
4. Creamos nuevo grupo de volúmenes VG
5. Creamos el volúmen lógico LV */newvar*
6. Ciframos
7. Montamos el */var* (llevamos el */var* a */newvar* una vez que está el sistema cifrado con LVM on LUKS)

<u>Arquitectura de la Lección3:</u>



De esta manera */var* estará completamente aislado del resto del sistema dado que hemos sacado un VG nuevo a parte del que nos proporciona la instalación por defecto de CentOS y con la información replicada gracias al RAID1.

Preparación del sistema descrito anteriormente:

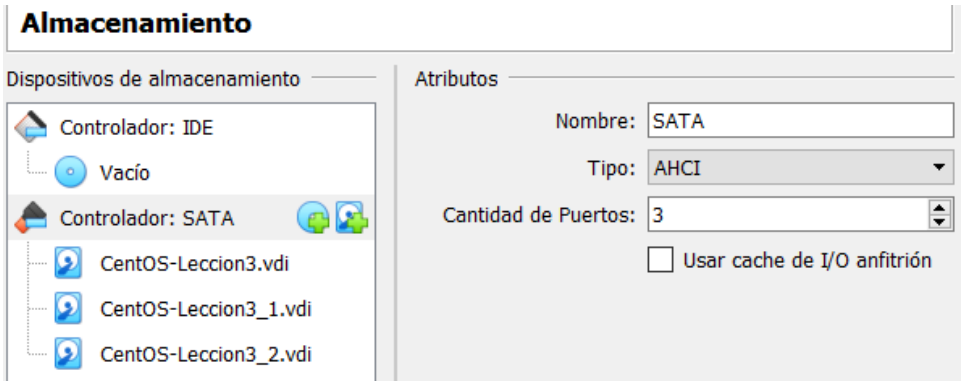
Como hemos visto anteriormente, tras la instalación por defecto de CentOS, nuestro sistema queda como sigue (punto de partida):

```
CentOS Linux 8 (Core)
Kernel 4.18.0-193.el8.x86_64 on an x86_64

Activate the web console with: systemctl enable --now cockpit.socket

localhost login: peroj
Password:
[peroj@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 1024M  0 rom
```

1º) Con la máquina apagada, añadimos los 2 discos duros (*sdb* y *sdc)* como siempre desde la *Configuración/Almacenamiento/Controlador:SATA/AgregarDiscosDuros*.



Arrancamos y comprobamos que se han insertado correctamente los discos con ***lsblk***.

```
[peroj@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
sdc           8:32   0    8G  0 disk
sr0          11:0    1 1024M  0 rom
```

Antes de continuar debemos instalar desde el gestor de paquetes ***yum***[4] (dado que CentOS viene sin ***snap*** ni ***apt***) la herramienta con la que crearemos el RAID (***mdadm*** es una herramienta que nos permite gestionar los RAID software en Linux) con ***yum install mdadm***. Previamente deberemos dar de alta la interfaz de red para tener conexión a intertet, ya que por defecto en CentOS viene desactivada. Podemos hacerlo con: ***sudo ifup enp0s3*** y lo comprobamos haciendo ping a google.

```
[peroj@localhost ~]$ sudo su
[root@localhost peroj]# ifup enp0s3
Conexión activada con éxito (ruta activa D-Bus: /org/freedesktop/NetworkManager/ActiveConnection/1)
[root@localhost peroj]# yum install mdadm
CentOS-8 - AppStream                                    5.5 MB/s | 5.8 MB     00:01
CentOS-8 - Base                                         2.6 MB/s | 2.2 MB     00:00
CentOS-8 - Extras                                        13 kB/s | 8.1 kB     00:00
El paquete mdadm-4.1-13.el8.x86_64 ya está instalado.
Dependencias resueltas.
Nada por hacer.
¡Listo!
```

2º) Creación RAID1 software:

Ejecutaremos el comando siguiente, donde ***--create /dev/md0*** indica que se creará el RAID con el nombre /dev/md0, ***--level=1*** indica el tipo de RAID, ***--raid-devices=2*** ya que tendremos dos discos duros dentro del RAID1 (*sdb* y *sdc)* y le indicamos la ruta de ambos discos duros.

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

Una vez ejecutado nos pregunta si meteremos un */boot* en este RAID, y si es así, que nos aseguremos que el *boot loader* es compatible y puede arrancar desde ahí. Pero nosotros no queremos meter ningún */boot* en este RAID, por lo que no tendremos que asegurarnos de esto (continuamos con ***y***). En segundo plano está sincronizando los dos discos duros del RAID1, por eso nos aparece el mensaje ***md: md0: resync done*** pasado un tiempo. Y comprobamos con ***lsblk***.

```
[root@localhost peroj]# mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sdb /dev/sdc
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?
Continue creating array? (y/n) y
mdadm: Defaulting to version 1.2 metadata
[ 2175.638644] md/raid1:md0: not clean -- starting background reconstruction
[ 2175.638685] md/raid1:md0: active with 2 out of 2 mirrors
[ 2175.638747] md0: detected capacity change from 0 to 8580497408
mdadm: array /dev/md0 started.
[ 2175.643345] md: resync of RAID array md0
[root@localhost peroj]#
[root@localhost peroj]#
[root@localhost peroj]#
[root@localhost peroj]# [ 2217.311749] md: md0: resync done.

[root@localhost peroj]# 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
└─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
```

[4] Con ***yum search*** podemos buscar las dependencias de la herramienta que queremos instalar. Por si nos interesa una versión o dependencia concreta.

3º) Crear el volumen físico PV del RAID *md0* para poder abstraer el RAID y que el LVM lo pueda usar.

pvcreate /dev/md0

Comprobamos con ***pvdisplay***.

También podemos comprobarlo con la herramienta ***pvs*** (show) que nos mostrará un resumen de los PV.

```
[root@localhost peroj]# pvcreate /dev/md0
  Physical volume "/dev/md0" successfully created.
[root@localhost peroj]# pvdisplay
  --- Physical volume ---
  PV 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               0Eo9BK-51b6-eWu9-96vs-zw0j-BWl1-a2C2lq

  "/dev/md0" is a new physical volume of "7,99 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/md0
  VG Name
  PV Size               7,99 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               eHsOFv-NjPT-Nsws-DksA-PVqf-lw1D-V553f6

[root@localhost peroj]# pvs
  PV         VG Fmt  Attr PSize  PFree
  /dev/md0      lvm2 ---   7,99g 7,99g
  /dev/sda2  cl lvm2 a--  <7,00g    0
```

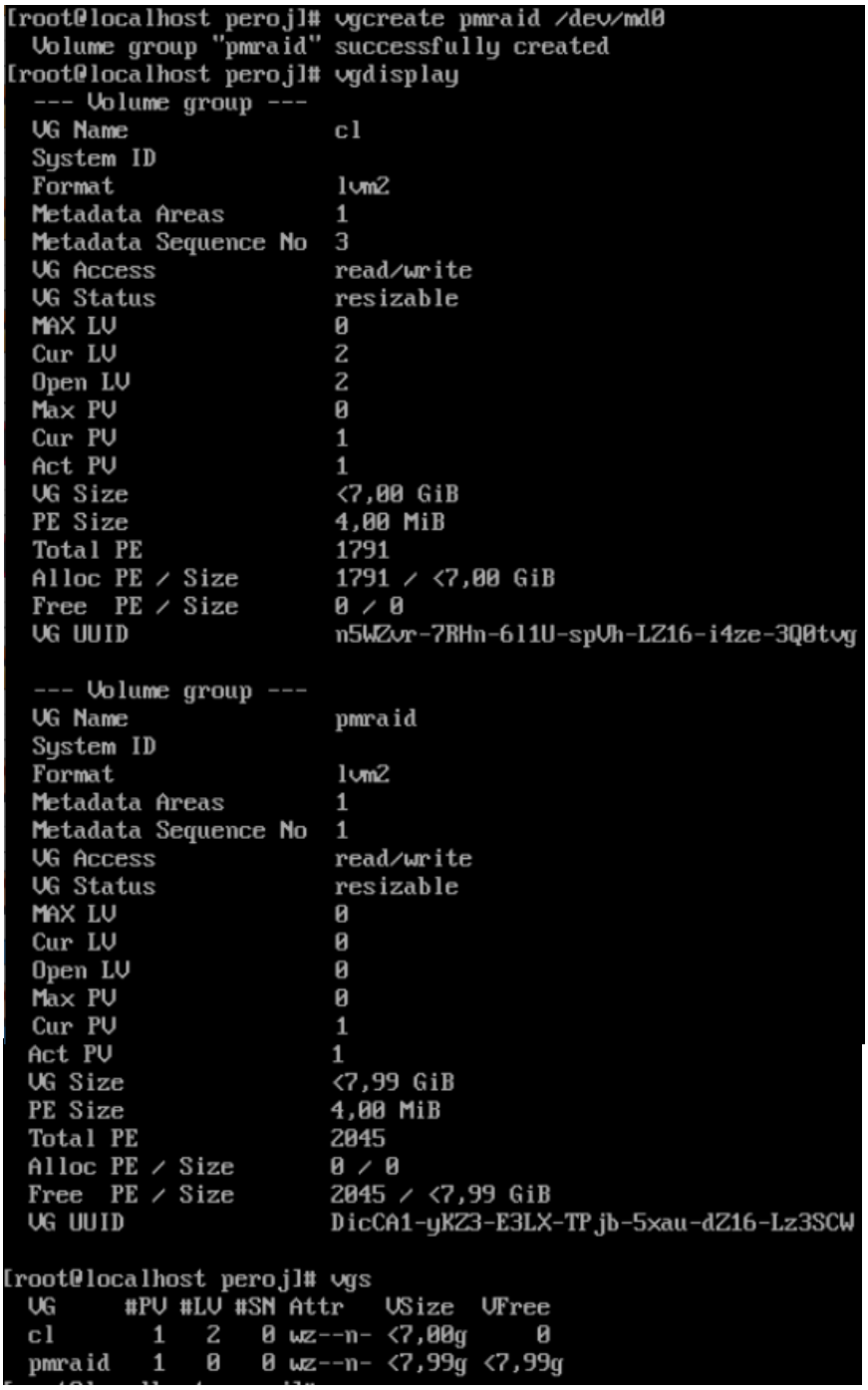
Como podemos ver el PV *md0* recién creado no pertenece a ningún grupo de volúmenes VG, ya que simplemente lo hemos abstraído, pero no está enganchado a ningún lado. Procedemos.

4º) Creamos nuevo grupo de volúmenes VG:

vgcreate pmraid /dev/md0 (vgcreate <nombre> <lista con los PV que queremos>)

Comprobamos con ***vgdisplay***

También podemos comprobarlo con la herramienta ***vgs*** (show) que nos mostrará un resumen de los VG.



Como podemos ver, tenemos el grupo *cl* que es el creado por CentOS, tiene un PV que es la abstracción de *sda2* y que contiene 2 LV (*/boot* y */swap*). Y tenemos el recién creado *pmraid* que tiene un PV que es *md0* pero no tiene ningún LV en este grupo nuevo. Continuamos.

5º) Creamos el volúmen lógico LV /*newvar* :

Crearemos el volúmen lógico /*newvar* al que luego le vamos a mover /var.

lvcreate -n newvar -L 1G pmraid (lvcreate -n <nombre> -L <tamaño> <VG al que va LV>)

Comprobamos con ***lvdisplay.***

También podemos comprobarlo con la herramienta ***lvs*** (show) que nos mostrará un resumen de los LV.

```
[root@localhost peroj]# lvcreate -n newvar -L 1G pmraid
  Logical volume "newvar" created.
[root@localhost peroj]# lvdisplay
  --- Logical volume ---
  LV Path                /dev/cl/swap
  LV Name                swap
  VG Name                cl
  LV UUID                Xo1GdC-M94j-UB68-PnNm-HYtK-zCJf-ln2uih
  LV Write Access        read/write
  LV Creation host, time localhost, 2020-10-23 06:11:09 -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                G397yl-Ceag-GoYc-IQ18-F6s5-Q86P-eqnaEN
  LV Write Access        read/write
  LV Creation host, time localhost, 2020-10-23 06:11:10 -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/pmraid/newvar
  LV Name                newvar
  VG Name                pmraid
  LV UUID                J20064-2Aqr-WwQP-yD9Z-HoPI-chZo-cd9Gbt
  LV Write Access        read/write
  LV Creation host, time localhost.localdomain, 2020-10-23 07:54:32 -0400
  LV Status              available
  # open                 0
  LV Size                1,00 GiB
  Current LE             256
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:2

[root@localhost peroj]# 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
  newvar pmraid -wi-a-----   1,00g
```

Vemos los LV del sistema, *swap* con 820 MB, *root* con 6’20 GB y *newvar* con 1 GB.

6º) Ciframos:

Módulo o estándar de encriptación ***LUKS*** (Linux Unified Key Setup) es un estándar que tienen los sistemas Linux para hacer el cifrado.

Hay dos tipos de cifrado principalmente:

- ***LVM on LUKS*** (en esta versión de CentOS y Ubuntu se sigue este)

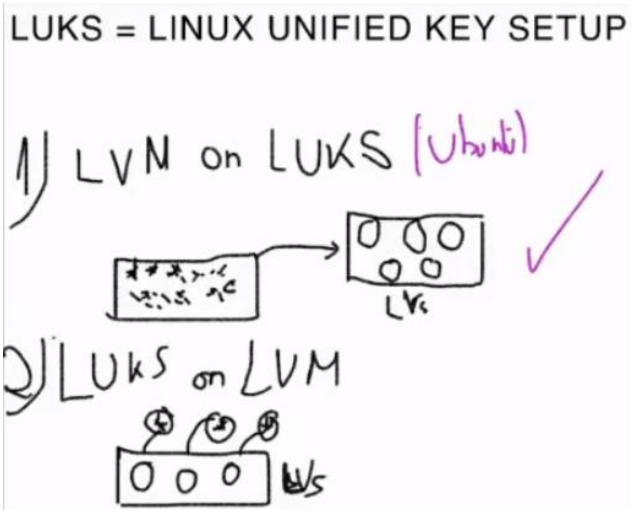
Tenemos la partición (contenido a cifrar). En este caso, primero cifra todo el contenido y una vez que está cifrado el contenido, coge y crea los volúmenes lógicos.

Menos carga computacional porque tiene que hacer menos operaciones, encripta y desencripta una única vez.

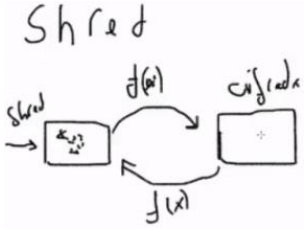
- ***LUKS on LVM***

La estrategia es la contraria. Primero creamos los volúmenes lógicos y una vez creados los cifrados cada uno por separado.

Mayor carga computacional porque tiene que encriptar y desencriptar tantas veces como volúmenes lógicos tengamos. Más seguro, ya que podemos cifrar por separado, es decir, cada LV puede llevar una contraseña distinta.



Cuando la información es muy sensible se suele usar la herramienta ***Shred***. Lo que hace es ofuscar[5] y desofuscar el contenido, fijaremos la f(x) que hemos aplicado a la información sin ofuscar y la aplicaremos la $f^{-1}(x)$ para llegar a la información.



[5] Ofuscar la información o código: Se suele hacer aplicando funciones matemáticas que se aplican al contenido y si no sabes la función inversa no podrás ver la información original.

Procedemos a hacer la encriptación *LVM on LUKS con la herramienta* ***cryptsetup*** , que maneja los volúmenes encriptados LUKS.

5.1 - Formatear con ***luksFormat*** para darle formato de encriptado a una partición (newvar en este caso):

cryptsetup luksFormat /dev/pmraid/newvar

```
[root@localhost peroj]# cryptsetup luksFormat /dev/pmraid/newvar

WARNING!
========
Esto sobreescribirá los datos en /dev/pmraid/newvar de forma irrevocable.

Are you sure? (Type uppercase yes): YES
Introduzca la frase contraseña de /dev/pmraid/newvar:
Verifique la frase contraseña:
[root@localhost peroj]#
```

5.2 - Abrir esa partición en un punto de montaje nuevo con ***luksOpen*** para poder entrar a los datos desencriptados. (usar con la nomenclatura *mapper* de acceso[6] a los volúmenes lógicos).

cryptsetup luksOpen /dev/mapper/pmraid-newvar pmraid-newvar_crypt

(<partición que queremos abrir> <nombre del punto de montaje donde queremos que se abra esta partición cifrada>)

Ejecutamos, introducimos la contraseña que pusimos al cifrar y vemos con ***lsblk*** que ya tendríamos accesible nuestro *newvar* encriptado.

```
[root@localhost peroj]# cryptsetup luksOpen /dev/mapper/pmraid-newvar pmraid-newvar_crypt
Introduzca la frase contraseña de /dev/mapper/pmraid-newvar:
[root@localhost peroj]# 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
└─md0                       9:0    0    8G  0 raid1
  └─pmraid-newvar         253:2    0    1G  0 lvm
    └─pmraid-newvar_crypt 253:3    0 1008M  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 1008M  0 crypt
sr0                        11:0    1 1024M  0 rom
[root@localhost peroj]#
```

Una vez abierto y siguiendo con LVM on LUKS, debemos llevarnos /var a /newvar.

[6] Nomenclaturas de acceso a los volúmenes lógicos: Cuando se inicia el sistema no está disponible la nomenclatura /dev/VG/LV, la que si está disponible es la del módulo mapper. Por eso esta última, se suele usar en el fichero fstab.

6º) Montamos el */var* (llevamos el */var* a */newvar* una vez que está el sistema cifrado con LVM on LUKS).

6.1) Ponemos el sistema en modo mantenimiento con ***systemctl isolate rescue*** .

```
[root@localhost ~]# systemctl status
● localhost.localdomain
    State: maintenance
     Jobs: 0 queued
   Failed: 0 units
    Since: Sat 2020-10-24 05:21:09 EDT; 26min ago
   CGroup: /
           ├─init.scope
           │ └─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 16
           └─system.slice
             ├─rngd.service
             │ └─843 /sbin/rngd -f --fill-watermark=0
             ├─systemd-udevd.service
             │ └─688 /usr/lib/systemd/systemd-udevd
             ├─systemd-journald.service
             │ └─662 /usr/lib/systemd/systemd-journald
             └─rescue.service
               ├─2320 /usr/lib/systemd/systemd-sulogin-shell rescue
               ├─2321 bash
               ├─2340 systemctl status
               └─2341 (pager)
```

6.2) Le damos formato a la particion: le damos formato a los datos para poder acceder a la información usando **mkfs -t ext4 /dev/mapper/pmraid-newvar_crypt** (la que cuenta es la *crypt,* la otra simplemente contiene los datos, la que se abre y con la que se trabaja es la *cypt*).

```
[root@localhost ~]# mkfs -t ext4 /dev/mapper/pmraid-newvar_crypt
mke2fs 1.45.4 (23-Sep-2019)
Se está creando un sistema de ficheros con 258048 bloques de 4k y 64512 nodos-i
UUID del sistema de ficheros: 86355e69-a32e-4c83-ba85-848de5deffb1
Respaldos del superbloque guardados en los bloques:
        32768, 98304, 163840, 229376

Reservando las tablas de grupo: hecho
Escribiendo las tablas de nodos-i: hecho
Creando el fichero de transacciones (4096 bloques): hecho
Escribiendo superbloques y la información contable del sistema de ficheros: hecho
```

6.3) Montamos esta partición en un punto de montaje del sistema (crear la carpeta temporal en */mnt*) para poder pasarle */var* a la partición (la copia atómica con ***-a***, que incluye ***-dR*** para copia recursiva de directorios y ***--preserve=all*** para la copia de los contextos).

```
[root@localhost ~]# mkdir /mnt/newvar
[root@localhost ~]# ls /mnt/
newvar
[root@localhost ~]# mount /dev/mapper/pmraid-newvar_crypt /mnt/newvar
[ 2018.679577] EXT4-fs (dm-3): mounted filesystem with ordered data mode. Opts: (null)
[root@localhost ~]# cp -a /var/. /mnt/newvar
```

Comprobamos que contienen exactamente lo mismo y se mantienen los contextos (*/var* y */mnt/newvar* con ***ls -Z***).

Ahora automatizamos esto modificando el fichero ***fstab*** para que en el arranque nos monte siempre /var en la partición /newvar_crypt).

```
/dev/mapper/pmraid-newvar_crypt /var                    ext4    defaults        0 0
```

Antes de montar de forma definitiva */var* en la partición crypt, vamos a mover el antiguo */var* a */var_OLD* para hacer un respaldo de seguridad.

Desmontamos la partición auxiliar ***umount /mnt/newvar*** y después lo montamos desde el fichero ***fstab*** con ***mount -a*** (parcheando previamente con ***mkdir /var*** y haciendo un ***restorecon /var*** para los contextos).

```
[root@localhost ~]# mv /var /var_OLD
[root@localhost ~]# ls /
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var_OLD
boot  etc  lib   media  opt  root  sbin  sys  usr
[root@localhost ~]# umount /mnt/newvar
[root@localhost ~]# mount -a
mount: /var: el punto de montaje no existe.
[root@localhost ~]# mkdir /var
[root@localhost ~]# ls -Z /
       system_u:object_r:bin_t:s0 bin           system_u:object_r:proc_t:s0 proc
      system_u:object_r:boot_t:s0 boot    system_u:object_r:admin_home_t:s0 root
    system_u:object_r:device_t:s0 dev        system_u:object_r:var_run_t:s0 run
       system_u:object_r:etc_t:s0 etc            system_u:object_r:bin_t:s0 sbin
 system_u:object_r:home_root_t:s0 home           system_u:object_r:var_t:s0 srv
       system_u:object_r:lib_t:s0 lib          system_u:object_r:sysfs_t:s0 sys
       system_u:object_r:lib_t:s0 lib64          system_u:object_r:tmp_t:s0 tmp
       system_u:object_r:mnt_t:s0 media          system_u:object_r:usr_t:s0 usr
       system_u:object_r:mnt_t:s0 mnt    unconfined_u:object_r:default_t:s0 var
       system_u:object_r:usr_t:s0 opt            system_u:object_r:var_t:s0 var_OLD
[root@localhost ~]# restorecon /var
[root@localhost ~]# mount -a
```

```
[ 3198.666369] EXT4-fs (dm-3): mounted filesystem with ordered data mode. Opts: (null)
[root@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
└─md0                       9:0    0    8G  0 raid1
  └─pmraid-newvar         253:2    0    1G  0 lvm
    └─pmraid-newvar_crypt 253:3    0 1008M  0 crypt /var
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 1008M  0 crypt /var
sr0                        11:0    1 1024M  0 rom
```

Ya nos hemos llevado */var* al RAID con los dos discos duros y está encriptado. Ahora solo falta modificar el fichero ***crypttab*** para que en el arranque del sistema nos desencripte la partición. El formato del fichero es una línea para cada partición a desencriptar: nombre de la partición donde se quiere montar la información desencriptada (este no es el nombre de la partición encriptada, es el nombre del acceso que nosotros le damos a la información cifrada *pmraid-newvar_crypt*), UUID=..... de la partición física encriptada *pmraid-newvar* y finalmente *none*. Con esto lo que hace el sistema cuando arranca es coger el UUID, busca qué es (la partición encriptada) y le hace un *luksOpen* para abrirla en el nombre (pmraid-newvar_crypt).

Para ver el UUID de las particiones usamos ***blkid***. Para añadirlo y sabiendo que es la única línea que contiene la palabra “crypto” podemos usar lo siguiente:

blkid | grep crypto >> /etc/crypttab

```
[root@localhost ~]# blkid
/dev/sda1: UUID="20aafe90-23d9-4d1d-ae23-7ef4ffc7a766" TYPE="ext4" PARTUUID="d106bf6d-01"
/dev/sda2: UUID="0Eo9BK-51b6-eWu9-96vs-zw0j-BW11-a2C2lq" TYPE="LVM2_member" PARTUUID="d106bf6d-02"
/dev/sdb: UUID="0914504c-1e7a-5880-4412-952cc3292ed0" UUID_SUB="70aaab0c-52b0-a0d5-5bb2-ea6247356d48
" LABEL="localhost.localdomain:0" TYPE="linux_raid_member"
/dev/sdc: UUID="0914504c-1e7a-5880-4412-952cc3292ed0" UUID_SUB="c83f87d0-a0aa-5884-26da-393b8085b041
" LABEL="localhost.localdomain:0" TYPE="linux_raid_member"
/dev/mapper/cl-root: UUID="5af4abb4-4cb0-44f2-95f8-7a8efa369553" TYPE="xfs"
/dev/mapper/cl-swap: UUID="6739b45b-01ca-46d0-a7b1-7faba134fc83" TYPE="swap"
/dev/md0: UUID="eHsOFv-NjPT-Nsws-DksA-PVqf-lw1D-V553f6" TYPE="LVM2_member"
/dev/mapper/pmraid-newvar: UUID="e422e6fa-6dfd-4491-8568-d46670bed387" TYPE="crypto_LUKS"
/dev/mapper/pmraid-newvar_crypt: UUID="86355e69-a32e-4c83-ba85-848de5deffb1" TYPE="ext4"
[root@localhost ~]# blkid | grep crypto >> /etc/crypttab
[root@localhost ~]# vi /etc/crypttab
```

```
pmraid-newvar_crypt     UUID=e422e6fa-6dfd-4491-8568-d46670bed387       none
```

Reiniciamos ***reboot*** y comprobamos que cuando arranca el sistema nos pide la contraseña, nos monta la partición y arranca todo correctamente.

```
[  OK  ] Found device /dev/disk/by-uuid/e422e6fa-6dfd-4491-8568-d46670bed387.
[  OK  ] Started LVM event activation on device 9:0.
         Starting Cryptography Setup for pmraid-newvar_crypt...
Please enter passphrase for disk pmraid-newvar (pmraid-newvar_crypt) on /var! [
 (sda1): mounted filesystem with ordered data mode. Opts: (null)
[   15.557605] snd_intel8x0 0000:00:05.0: white list rate for 1028:0177 is 48000
**************
```

```
[peroj@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
└─md0                     9:0    0    8G  0 raid1
  └─pmraid-newvar       253:2    0    1G  0 lvm
    └─pmraid-newvar_crypt 253:3  0 1008M  0 crypt /var
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 1008M  0 crypt /var
sr0                      11:0    1 1024M  0 rom
[peroj@localhost ~]$ _
```