Universidad de Granada

ISE

2018

Cuaderno de prácticas de ingeniería de servidores

Autor:
Francisco Navarro Morales

Índice

1. Práctica 1

1.1. Primera sesión

1.1.1. Introducción a los servidores.

1.1.2. Configuración de ubuntu server.

A continuación describiré el proceso para configurar ubuntu server en una máquina virtual (VirtualBox).

La primera pregunta que nos haremos será cuanta memoria RAM asignaremos a la máquina virtual. Ubuntu server podría funcionar con tan solo 500mb (sería una cantidad mínima a asignar) y, en caso de tratarse de un servidor que fuera a tener mucha carga, le podríamos asignar varios gigabytes. En mi caso voy a dejar la opción por defecto (1024 mb) ya que no sé que carga va a tener a priori la máquina y tampoco necesito optimizar los recursos hasta el punto de ahorrar en memoria RAM.

El tipo de disco también seleccionaré la opción por defecto de virtualbox.

Respecto al tipo de almacenamiento (**dinámico vs estático)**) tenemos que tener en cuenta que el almacenamiento dinámico ofrece más versatilidad pero también produce una mayor fragmentación. Es decir, utilizar un tamaño fijo nos hará ocupar recursos innecesariamente desde un principio pero dará mejor rendimiento (sería una manera de optimizar en caso de saber de manera más o menos exacta las necesidades de memoria del servidor). En este caso utilizaremos reserva dinámica porque la fragmentación no es algo que nos deba preocupar para el desarrollo de las practicas.

Una vez creada la máquina virtual, vamos a añadir el ISO de instalación de ubuntu server y además vamos a añadir un segundo disco puesto que queremos utilizar tecnología RAID (redundant array independent disc). En concreto vamos a hacer un raid de tipo 1, que consiste en hacer un mirror (una copia exacta) de los datos en dos o más discos a la vez. Este tipo de configuración mejora los tiempos de lectura (aunque puede suponer empeorar el tiempo de escritura) y aporta mayor seguridad a la información almacenada (si un disco falla, tenemos el resto).

Para configurar los discos, click derecho en la máquina virtual, settings¿storage, en el controlador IDE añadiremos el ISO con ubuntu server y en los controladores SATA crearemos un nuevo disco (click en controller: SATA ¿botón cuadrado con

un +)

Iniciamos la máquina virtual, idioma inglés, install ubuntu server, seleccionamos el idioma, la configuración de teclado en español y rellenamos algunos valores como el nombre del host o del usuario principal.

Cuando nos pregunten si queremos encriptar el directorio home le decimos que no, puesto que lo vamos a encriptar manualmente junto con el resto del sistema.

1.1.3. Particionado para ubuntu server

El procedimiento es el siguiente:

- creamos la tabla de particiones para cada uno de los discos
- creamos el MD (multiple device-drive aka Linux software RAID)
- creamos el logic volume y las particiones de este
- creamos los volúmenes de cifrado
- formateamos y asignamos puntos de montaje a los volúmenes cifrados.

En el momento de realizar el particionamiento seleccionamos 'Particionado manual' y ahora seleccionamos cada uno de los discos y le decimos al asistente que cree una tabla de particiones en él.

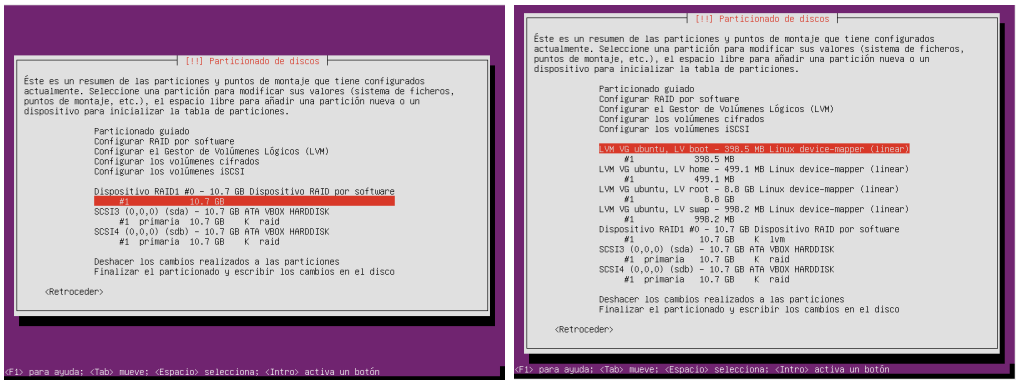

Figura 1: Visualización del esquema de particionado justo después de crear el volumen físico, md0, y justo después de configurar el LVM y crear las particiones.

Una vez creadas las particiones procederemos a crear un nuevo dispositivo md (raid por software). Se nos preguntará qué queremos hacer, le indicamos que crear un nuevo dispositivo MD, seleccionamos raid tipo 1 y le especificamos que utilizará dos discos, seleccionamos los dos discos con espacio y pulsamos enter para proceder. Finalmente pulsamos en Terminar. (ver figura 1).

Ahora tenemos definido un volumen físicos (PV, physical volumen) y en él, definiremos 4 volúmenes lógicos (LV, logical volume); root, boot, home y swap.

Para ello, crearemos el LVM (logical volume manager). Vamos a configuración de LVM, crear grupo de volúmenes (lo llamaremos ubuntu) y seleccinamos el dispositivo físico md0.

Ahora el procedimiento es el siguiente: dado que cifraremos los volúmenes lógicos por separado (más seguro y versátil que cifrar todo el LVM) y que, al cifrar un volumen lógico se pierde la información de este referente a aspectos como el formato o el punto de montaje...

Por tanto, una vez creado el LVM, seleccionaremos çrear volumen lógico"para crear cada nueva partición.

El orden no importa, pero dejaremos el root para el final para asegurarnos de asignarle toda la memoria posible. Respecto a cuanta memoria asignar: para boot bastaría con unos 200 mb, pero como tampoco andamos mal de memoria podemos asignar hasta unos 400 mb... En un pc asignaríamos bastante más memoria al home pero al tratarse de un servidor... no tendremos muchos datos en esta partición así que 500 mb está bien. Para swap podemos utilizar un gb aunque también estaría bien un poco menos (o un poco más pero no hay que hacerla demasiado grande). Respecto al formato... (que no deberíamos aplicar todavía puesto que de hacerlo se perdería al cifrar las particiones) swap será memoria de intercambio y las otras tres pueden ser ext4 o ext2, preferimos ext4 porque es más moderna y en teoría mejor. Una posible excepción sería el boot al que le pondremos ext2 (si queremos) porque las ventajas que aporta ext4 no se van a notar y ext2 es más 'ligero'. Como ya se ha especificado, al crear la partición para root (la última) asignamos todo el espacio restante (el que aparece por defecto). En la figura 1 se puede visualizar como debe quedar el esquema hasta el momento.

A continuación seleccionaríamos 'configurar los volúmenes de cifrado' y le damos a crear volúmenes de cifrado. Seleccionamos todas las particiones excepto el boot (se podría cifrar pero es más complicado). Las dejamos como están (opciones

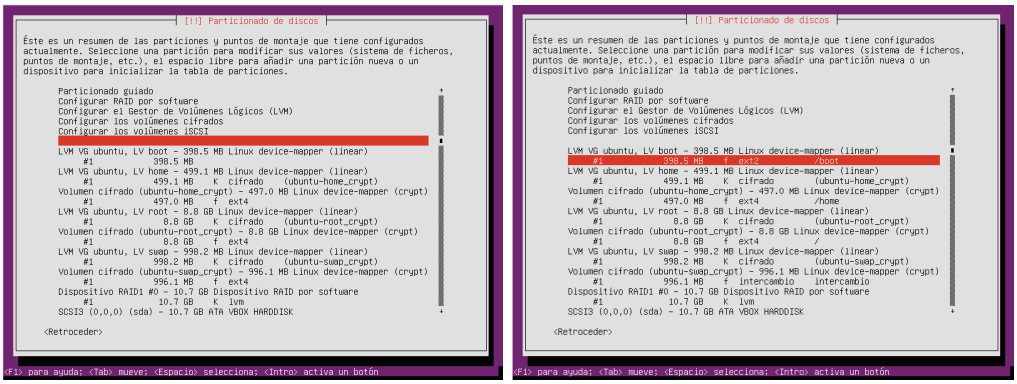

Figura 2: Visualización del esquema de particionado justo después de cifrar el sistema de archivos y justo después de asignar a las particiones formato y puntos de montaje.

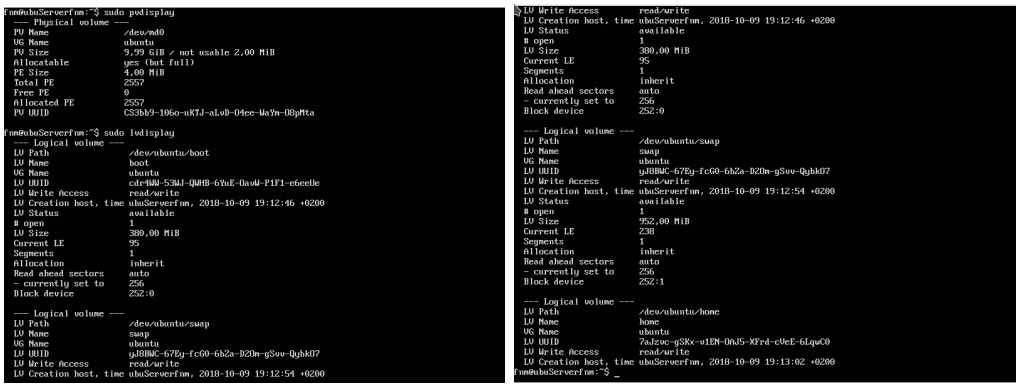

Figura 3: Comprobaciones post-instalación con pvdisplay y lvdisplay.

por defecto) y procedemos activando los volúmenes cifrados.

Debemos configurar las distintas particiones hasta pasar del esquema de la figura 3 al de la figura 3

Una vez esté todo como en la figura 3 podremos proceder pulsando 'Finalizar el particionado y escribir los cambios en el disco'.

1.2. Segunda sesión.

En esta parte trabajaremos con una imagen de centOS con el siguiente escenario: un cliente nos pide alojamiento para una web de vídeos y, debido a esto, /var se nos queda sin espacio. Entonces compramos un nuevo disco (sdb, en la práctica

será un disco virtual, simulado) y queremos mover /var a un nuevo volumen lógico con mayor tamaño.

La dinámica será la siguiente:

- creamos un nuevo volumen físico
- extendemos el LVM para añadir el nuevo disco.
- creamos una nueva partición o volumen lógico que será el nuevo /var
- creamos el sistema de archivos en la nueva partición
- montamos la partición en un nuevo punto de montaje
- copiamos toda la información del /var previo
- asignamos el punto de montaje /var a la nueva partición
- liberamos el espacio del antiguo /var

Para crear el volumen físico utilizaremos **pvcreate /dev/sdb** y comprobamos que se haya creado correctamente con **pvs**. ¿Haría falta crear la tabla de partición en el dispositivo? NO, no es necesario, podemos crearlo sin más.

A continuación extenderíamos el LVM con **vgextend cl /dev/sdb** (cl es el nombre que le da centOS al LVM principal) y comprobamos que todo está bien con vgs.

Creamos un nuevo volumen lógico con **lvcreate -L 4G -n newvar cl** y comprobamos que todo está correcto con **lvdislay**. Los volúmenes lógicos tienen la pinta **/dev/cl/newvar** o bien **/dev/mapper/vgn-lua**.

Ahora creamos el sistema de archivos con **mkfs -t ext4 /dev/cl/newvar**, aquí podríamos valorar algunas opciones como por ejemplo cambiar el tamaño de bloque para obtener un mejor rendimiento en ciertos casos.

Creamos un punto de mntaje y montamos el volumen lógico: **mkdir /media/-newvar && mount /dev/cl/newwvar /media/newvar)**

A continuación copiamos el var antiguo con **cp -a /var/. /media/newvar**. Un detalle importante es que deberíamos asegurar la atomicidad al copiar. Si copiamos cantidades enormes de datos de golpe podrían darse problemas por culpa del uso del servidor a la vez que se copian los datos. Una solución sería cortar el acceso al

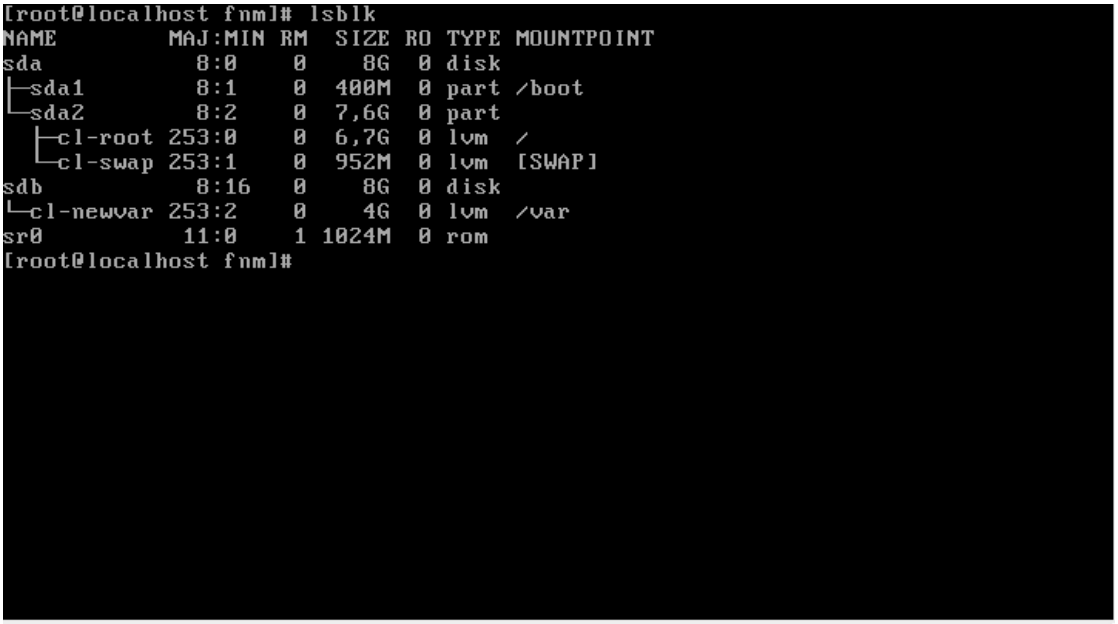

Figura 4: Output de lsblk tras completar todos los pasos

servidor mientras se hace la copia o (preferiblemente) cambiar el nivel de ejecución (targets de systemd) para asegurar que la copia se realiza con preferencia sobre cualquier otra acción: **systemctl isolate runlevel1.target**

A continuación tendríamos que editar el archivo **etc/fstab** y añadir la línea **'/dev/mapper/cl-newvar /var ext4 defaults 00'**

Hacemos los cambios efectivos con **mount -a**, comprobamos que todo es correcto y, finalmente, procedemos a liberar el espacio del antiguo /var

Para ello tenemos que volver a la configuración original desmotando el nuevo /var (**umount /dev/mapper/cl-var**), esto hará que se vuelva a montar el antiguo var en /var. **lsblk** para comprobarlo. A continuación **mv /var /media/-varold** (por si acaso), **mkdir /var**, **restorecon /var** y finalmente **mount -a**

Y ya estaría.

1.3. Tercera sesión.

En esta última sesión de la primera práctica vamos a hacer un RAID 1 para el var y encriptarlo.

Puedes ir usando lsblk para comprobar el esquema de dispositivos del sistema.

Para crear el RAID utilizaremos el comando mdadm (no viene instalado por defecto). Al abrir la máquina virtual no tendremos internet... hay que levantar la interfaz de red. Buscamos el nombre de la interfaz con **ip addr** y **ifup *interfaz*** para levantar la interfaz.

Comprobamos que tenemos internet con un ping y usamos yum para instalar mdadm **yum install mdadm**.

ahora el comando se usa tal que: **mdadm –create /dev/md0 –level=1 -raid-devices=2 /dev/sdX /dev/sdY** (los dispositivos los hemos comprobado con lsblk)

ahora creamos el volumen físico en el nuevo raid: **pvcreate /dev/md0**

En este caso no nos vale con extender el viejo grupo de volúmenes... necesitamos uno nuevo para evitar que se siga escribiendo en el antiguo var hasta llenarlo.

vgcreate pmraid1 /dev/md0 (comprobamos con vgs).

Ahora sí, creamos el volumen lógico **lvcreate -L 1G -n newvar pmraid1** y comprobamos con lvdisplay.

mkfs -t ext4 /dev/mapper/pmraid1-newvar

mkdir /media/newvar && mount /dev/mapper/pmraid1-newvar /media/newvar && cp -a /var/. /media/newvar

añadimos al /etf/fstab :

/dev/mapper/pmraid1-newvar /var ext4 defaults 0 0

y procedemos como en la sesión anterior:

Hacemos los cambios efectivos con **mount -a**, comprobamos que todo es correcto y, finalmente, procedemos a liberar el espacio del antiguo /var

Para ello tenemos que volver a la configuración original desmotando el nuevo /var (**umount /dev/mapper/pmraid-newvar**), esto hará que se vuelva a montar el antiguo var en /var. **lsblk** para comprobarlo. A continuación **mv /var /media/varold** (por si acaso), **mkdir /var**, **restorecon /var** y finalmente **mount -a**

Y ya estaría.

Finalmente nos tocaría cifrar el nuevo var. Instalamos la utilidad cryptsetup con yum install... y, como al cifrar se borra todo... primero vamos a hacer una copia **mkdir /media/varRAID/ && cp -a /var/. /media/varRAID**

antes de formatear hay que desmontar el dispositivo: **umount /dev/mapper/pmraid1-**

newvar pero es posible que no nos deje porque haya alguien utilizándolo... usando lsof (hay que instalarlo) en /var veríamos que dhcclient está escribiendo en var... podemos parar el proceso de dhclient con un kill -9 y su pid y luego probar a desmontar de nuevo.

Una vez desmontado... usamos **cryptsetup luksFormat /dev/mapper/pmraid1-newvar**

Como al encriptar se pierde el sistema de archivos, tendremos que crearlo otra vez. Pero primero hay que 'abrir' el sistema cifrado (algo parecido a montar un volumen lógico) con **cryptsetp luksOpen /dev/mapper/pmraid1-newvar pmraid1-newvar_crypt**

Aqui sería interesante usar el comando **shred** para meter basura en el volumen antes de cifrarlo para evitar posibles intentos de descifrar el volumen.

mkfs -t ext4 /dev/mapper/pmraid1-newvar_crypt

mkdir /media/newvar_crypt && mount /dev/mapper/pmraid1-newvar_crypt /media/newvar_crypt

cp -a /media/varRAID/. /media/newvar_crypt

editamos el fstab

/dev/mapper/pmraid1-newvar_crypt /var ext4 defaults 0 0

y el /etyc/crypttab, para el cual necesitamos el UUID del dispositivo... podemos utilizar blkid:

blkid | grep crypto >> /etc/crypttab

y luego editar con vim hasta que quede algo así:

pmraid1-newvar_crypt /dev/md0 UUID=793427491729387129873273 none

IMPORTANTE: RECUERDA AÑADIR _crypt al nombre!!

2. Práctica 2

2.1. Primera sesión: SSH

2.1.1. Instalar SSH

- **Ubuntu**

  Utilizaremos un sistema ubuntu server y otro centOS (se puede hacer uso del centOS por defecto o utilizar alguna las máquinas que se hicieron en la

primera practica).

SSH son las siglas de **Secure SHell**

Cuando un usuario trabaja con un servidor que utiliza el protocolo ssh utiliza **un cliente ssh**, el servidor ofrece un **servicio ssh**. Es decir, el protocolo tiene una estructura cliente servidor en la que el cliente hace peticiones al servidor. Cuando hablamos de SSH por tanto podemos estar refiriéndonos tanto al cliente SSH como al servidor SSH.

En ubuntu server podemos instalar el paquete **open-ssh** directamente con apt o bien utilizar la interfaz de instalación de ubuntuserver con el comando **tasksel**?. Preferiblemente no utilizaremos ese tipo de herramientas y usemos apt directamente porque nos dará más flexibilidad...

El "paquete.[en] realidad vien dividido en dos, **openssh-client** y **openssh-server**.

para ver procesos relacionados con ssh **pf -Af — grep ssh** tras instalar openssh veremos que ya se está ejecutando **sshd** la **d** viene de daemon y se refiere a que es un programa que se ejecuta en segundo plano (el que típicamente se ejecutaría en el servidor), para ver el estado y trabajar con el servicio utilizaremos systemctl:

systemctl status ssh[.service] o **systemctl start ssh[.service]** o stop, restart...

- **CentOS**

  En este caso openSSH viene instalado por defecto.

2.1.2. Configurar puerto (cambiarlo).

¿Archivo de configuración de ssh? en Ubuntu tenemos **/etc/ssh/sshd_config** (para el servicio) y **/etc/ssh/ssh_config** (atento a la d)

Usamos **su vi /etc/ssh/sshd_conf** y simplemente cambiamos el valor de **Port** a (por ejemplo 22022) y comprobamos que el cambio ha surtido efecto con el cliente ssh:

ssh localhost -p 22022

Esto fallará porque hay que reiniciar el servicio para que la edición surta efecto.

sudo systemctl restart sshd.service

sudo systemctl status sshd.service (aqui se puede ver que ahora ya sí está escuchando el puerto 22022)

ssh localhost -p 22022

Para probarlo desde el exterior (desde la otra maquina virtual o desde el host) hacemos

ssh 192.168.56.105 -p 22022 -l fnm -v

2.2. Denegar acceso root.

¿Por qué? Seguridad. En todos los sistemas hay un root y sería relativamente sencillo atacar un equipo probando contraseñas al azar para el root (más aún si ssh está en el puerto por defecto).

ubuntu archivo de configuración parámetro PermitRootLogin -¿prohibit-password lo cambiamos a **no** porque ubuntu por defecto te permite acceder como root con ciertas medidas de seguridad.. de esta manera aseguramos que no se puede de ningún modo.

2.3. Acceso sin contraseña.

utilizando un par de llaves pública/privada. id_rsa e id_rsa.pub una misma llave privada puede abrir varias 'cerraduras' públicas.

Para generar un par pública/privada usamos ssh-keygen. Dinámica:

En centOS (como cliente):

ssh-keygen

y luego usamos

ssh-copy_id 192-168-56105@fnm -p 22022 (recuerda que root me van a bloquear)

Si en ubuntu quisiera denegar el conectarse con contraseña una vez enviada la clave pública:

PasswordAutentication no

Así solo se podría acceder sin contraseña (con clave privada).

2.4. Segunda sesión: ssh en centOS, copias de seguridad y control de versiones.

en centOS la instalación de ssh viene realizada por defecto. ¿Cómo podríamos comprobar que el servicio está funcionando? **systemctl status sshd** en la última línea podríamos ver en qué puerto estamos escuchando.

2.4.1. Modificar el puerto de escucha

utilizaremos esta vez **sed** en lugar de vi.

sed -e "s/Port 22/Port 22022/" -i /etc/ssh/sshd_config

comprobamos **cat /etc/ssh/sshd_config — grep "Port 2"**

MUY IMPORTANTE: modificar el archivo de configuración de sshd no altera su funcionamiento: es necesario reiniciar el servicio.

Además de esto, resulta que en centOS el archivo de configuración está comentado con las opciones por defecto. Es decir, que tendremos que des-comentar la linea...

Además de esto, tendremos problemas porque SELinux nos va a impedir el cambio de puerto por temas de seguridad. Si leemos el archivo de configuración de SSH veremos que justo encima de Port .... pone qué debemos hacer para pedirle a SELinux que nos deje ser felices con nuestro puerto.

Necesitamos instalar **semanage**

Si no sabemos qué paquete necesitamos instalar para un comando podemos usar **yum provides semanage**. Veremos que necesitamos el paquete:

yum install policycoreutils-python

Cosas de semanage:

semanage port -l — grep ssh para listar puertos

semanage port -a -p tcp -t ssh_port_t 22022 (luego reiniciamos y ya estaría)

Otra diferencia entre ubuntu y centOS es que en ubuntu el firewall no viene activado por defecto mientras que en centOS sí, de forma que las peticiones externas de shh al servidor son bloqueadas aún después de haber hecho todos los cambios necesarios.

Para utilizar el firewal... saber que ubuntu usa uncomplicated firewall (ufw) y lo podríamos activar con ufw enable. Para permitir que se utilice el ssh tendríamos

que hacer ufw allow 22022.

En centOS tenemos firewall-cmd.

firewall-cmd –permanent –ad-port=443/tcp hará que en un futuro (cada vez que se reinicie el sistema) el firewall no impida el acceso a ssh al puerto 22022.

Si ejecutamos

firewall-cmd –ad-port=433/tcp esto permitirá el acceso en ese momento pero la próxima vez que reiniciemos ya no tendrá acceso (el cambio no es permanente).

Conclusión: usamos los dos o usamos el primero y reiniciamos el firewall con **firewall-cmd –reload**

2.4.2. Copias de seguridad y control de versiones

¿sería **cp -a /var/. /varOLD** una copia de seguridad? Sería conveniente guardar la fecha en que se hizo la copia: (añadimos el comando date con comillas simples y el formato que queremos) lo que hace una copia de seguridad como tal al final son los metadatos. La información sobre la copia de seguridad.

Puntos interesantes:

- Preferiblemente copiaremos en **otro disco físico** para que en caso de que se rompa el disco tengamos la otra copia.
- Si nos lo llevamos muy lejos (véase a Canadá) tendríamos problemas con la GDPR esa (ley de protección de datos). Además de que a la hora de recuperar los datos tardaríamos más de lo deseable. Sería considerable tener varias capas de localización de las copias. En un mismo computador, RAID 1, luego en el mismo edificio tendríamos otra capa de copias cada X tiempo (corto, por ejemplo diariamente) y finalmente un CPD (copia externa y lejos).
- deberíamos hacer las copias de seguridad incrementales.
- ¿compresión de los datos? quizá pero solo si de verdad nos rentara... si vamos a hacer una copia de seguridad y va a ser prácticamente igual de grande que los datos y encima nos va a saturar el disco, pues mejor no.
- a la hora de transferir los datos deberíamos usar **ssh**

- Tenemos que tener cuidado con que nadie nos modifique nada mientras se hace la copia... LVM nos da una solución para esto. Ya que tiene los **snapshots**, instantaneas del volumen lógico que queramos y poder hacer la copia de seguridad del snapshot mientras el sistema con el volumen real sigue funcionando.

- comandos navaja suiza: **dd**, lo que tenemos es lo que copiamos (bit a bit); tar para comprimir importantes opciones c para crear v para verbose, z para comprimir y f para indicarel nombre del archivo, -x para dexcomprimir. ; rsync que sirve para hacer cosas guays de sincronización, ojo a la opción –delete que borra lo que haya cambiado de una a la otra. Ojo con la barra punto y asterisco al final... revisarlo. aprendete lo de aviz. Y mira las transparencias y haz pruebas y esas cosas. ;

- alojomora

2.5. Tercera sesión: servidor web

Un servidor web utiliza el protocolo HTTP para dar respuesta a peticiones de clientes (devuelve documentos HTML) normalmente opera en el puerto 80 (por defecto).

HTML lo inventó el señor Tim Berners-Lee del cern en 1989.

HTML, JS ... estático...

¿dinamismo? -¿PHP y SQL

Apache vs nginx vs Ligthttpd vs IIS

lógica -¿PHP, python, perl, .net (asp), ¿por qué interpretados? -¿portabilidad? si pero... más bien porque puedes hacer cambios rápidamente sin recompilar.

LAMP STACK -¿Linux Apache MySQL PHP , otras combinaciones? pues otros STACKS ;D

framework -¿biblioteca de bibliotecas :D

2.5.1. AMO A INSTALAR LAMP EN UBUNTU

utilizamos tasksel y seleccionamos LAMP y ale ya estaría.

Comprobamos que esta todo ok: **systemctl status apache2.service** verás que el servicio está habilitado y activo. (habilitado es que cuando reinicie se va a activar por systemd) activo es que está operativo.

curl localhost para pedirnos una página a nosotros mismos... nos devolverá una página HTML osea que todo furula

php -a para probar el interprete de PHP

para probar mysql **systemctl status mysql** y para conectarnos **mysql -u root -p** (p de password)

2.6. AMO A INSTALAR LAMP EN CENTOS

yum search apache

yum install httpd

sum search php

yum install php

yum install mariadb

¿cuanta diferencia habría de espacio entre la instalación del LAMP server en centOS y Ubuntu?

systemctl status httpd por defecto desabilitado y desactivado! -¿**systemctl enable httpd && systemctl start httpd**

cuidado con mariaDB, hay un cliente y un servidor... necesitamos instalar y activar el SERVIDOR

yum install mariadb-server estará desactivado y deshabilitado por defecto **systemctl enable mariadb && systemctl start mariadb**

mariadb no nos ha preguntado la contraseña de root :D -¿el root no tiene contraseña por defecto.

mysql_secure_instalation quitamos usuarios anónimos, si, desactivamos el acceso remoto de root, ofcourse, eliminar la base de datos de test, por supuesto. Reload previleges? claro. Y ponemos contraseña al root, obviamente.

Si ahora mismo un cliente intentara hacer una petición al servidor tendría un error...

CentOS tiene el puerto 80 cerrado por defecto **firewall-cmd –ad-port=80/tcp –permanent** y luego **firewall-cmd –ad-port=80/tcp**

2.6.1. Añadir alguna lógica al servidor para conectar PHP y MySQL

cd /var/www/html

vi index.php

añadimos

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_close($link);
?>
```

editamos **/etc/httpd/conf.d/php.conf**

añadimos *.php a DirectoryIndex

systemctl restart httpd

Si sique sin funcionar... probamos a ejecutar el script directamente con php y veremos que nos falta x librería la instalamos (php-mysql) y a ver si furula... debería

El SElinux estará jodiendo la marrana y no nos va a dejar por ejemplo llamar al servidor con curl... habría que activar uno de los booleanos de configuracion (can_network_connect_db)

getsebool -a — grep httpd

setsebool -P httpd can_network_connect_db on

después de esto aún hay que reiniciar el servicio... fuck **systemctl restart httpd**

si creas un script en tu home y lo copias en el directorio... no te va a funcionar por el contexto del archivo... va a quedarse como que has creado el archivo en el home... y no lo va a aceptar SElinux. restorecon o creamos el script directamente en su sitio.

Cosas que hay que saber:

- fail2ban —— para evitar ataques de fuerza bruta en ssh o http o donde le diga.. **yum install epel-release** para poder instalar (**yum install fail2ban**) se activaría, se habilitaria (enable) y luego se modificaría el archivo de configuración o con fail2ban-client.... status para ver el estado (nos dirá que no hay) y otros probablemente para edita... pero vamos que podemos ir a **/etc/fail2ban/jail.conf**... en el mismo archivo veremos que no se debería editar ese archivo porque se sobreescribe al actualizar el paquete... debemos hacer una copia, **cp -a jail.conf jail.local** y luego **vi /etc/fail2ban/jail.local** si te dejas un espacio falla.... viendo el archivo se puede ver (en los comentarios) como va la movida.... después de modificarlo **systemctl restart fail2ban** y luegos puedes probar a loguearte veinte veces a ver qué pasa. para unbanear **fail2ban-client set sshd unbanip 192.168.56.1**

- tmux & screen —— son dos utilidades que hacen lo mismo (screen es más vieja y está más testada pero tmux es chachi tambien)

- RootKit Hunter (rkhunter) ——- esto va a cazar unos tipos de malware y otras movidas... hay que ejecutarlo de forma periódica... maybe con cron o con un timer.

Habría que mirarse los mensajicos del profe en el SWAD :D