

LECCION 1

Cortafuegos: Igual que los RAID que tienen HW y SW. Hacen que una máquina dedicada a dejar caer ciertos paquetes en base a un análisis o programación o un software que los monitorice y los deje caer según una condición. Uno de los firewall es hipertable (¿) que establece una cadena de reglas, es relativamente complejo y se sale un poco de la asignatura. Servidores web de altas prestaciones si llegan a configurarlo.

Para los que nunca han usado git, se explican dos o tres cosas y los resultados de la practica fallan en los exámenes.

YUM (REDHAT, **CENTOS**, FEDORA) y APT(DEBIAN, **UBUNTU**,...): Aplicaciones que nos permiten conectarnos con un servidor, ver la lista de paquetes que tiene y descargarlos de ahí.

Hacemos yum search para buscar qué tenemos. No ponemos interfaces graficas en nuestros servidores porque bajan el rendimiento, es un programa en ejecución y es mas seguro tener menos cosas en ejecución y porque las maquinas no van a tener pantalla, sino que vamos a hacer ssh desde casa o el trabajo, de forma remota para acceder al cluster.

Yum redirige a DNF, la evolución de yum (que tiene por debajo dnf)

En la rama DEBIAN tenemos apt. Se empezó a trabajar en los paquetes que son snap y Canonical o Red Hat querían converger a los snap. Los snap llevan las dependencias incorporadas, con lo cual es menos caotico de cara a gestionar esas dependencias sin resolver. Red Hat y Fedora siguieron con dnf y Canonical y Ubuntu usan de forma oficial los snaps.

SSH:

Viene de Secure Shell. Cuando nos conectábamos de una maquina a un servidor utilizábamos telnet y con wireshark veremos que la info que viaja es con el usuario y la contraseña en texto plano. Entonces, un alumno detecto la vulnerabilidad e implanto por seguridad el ssh. En centos no lo pregunta, directamente lo instala.

El cliente es ssh e inicia la conexión y nuestro servidor ejecuta un servicio que es sshd, que no es lo mismo. Son diferentes /etc/sshd_config y /etc/ssh_config.

Cosas importantes a configurar sobre seguridad. Root es el usuario de todas las maquinas, asi que pruebo a hacer root en la opción de login:

```
Ssh IP -l root
```

Resulta que centos tiene el permit root login puesto a true. En Ubuntu:

Ssh IP -I root

No te deja porque root no tiene contraseña en Ubuntu. Por ello vamos a editar con VI en Ubuntu. Si pones contraseña con "sudo passwd root" igualmente no te dejaría.

Los servicios



Con sshd nos conectamos al puerto (el puerto del que escucha) es el 22. Podemos cambiar el puerto para que nadie se conecte. Lo cambiamos con "sed"

COSAS QUE HACE EN MAQUINAS VIRTUALES:

CENTOS:

- Yum search ssh
 - o Aparecen las coincidencias en nombre
 - o También "dnf search ssh"
- Yum provides -> quien te proporciona el archivo
 - o Por ejemplo "yum provides libc.so.6"
- Yum provides libmpfr.so.4
 - o Te dice que viene de base
- Yum provides mc
 - o Mc-1:4.8.19.....
- Ps -Af | grep ssh
 - o Nos encontramos un proceso de sshd que esta como demonio
- Ssh localhost
 - o Accedes a tu propia maquina. Te dice que es la primera vez que te vas a conectar a ti mismo
 - o Yes
 - o Practicas, ISE
 - o Con eso te logeas
 - o Ctrl+d -> te sales
- Cd .ssh -> y te aparece dentro "known_hosts"
 - o Estará localhost y su fingerprint si le haces un cat
- Vi /etc/ssh/sshd_config

- Vemos que permit root login esta a yes
 - Lo pongo a que no, escape y :wp
- Vamos a probar que lo he restringido
 - Ssh IP -l root
 - Vemos que si te deja porque no ha reiniciado el servicio.
- Systemctl restart sshd.service
- Porbamos de nuevo haberlo restringido y ya no nos deja.
 - Para asegurarnos probamos "Ssh IP -l root -v"
 - No te deja
- Sed s/"Port 22"/"Port 22322"/ -i /etc/ssh/sshd_config
- Lo comprobamos
 - Ssh IP -p 22322
 - Fallo porque no ha hecho "systemctl restart sshd"
 - Comprobamos de nuevo tras ello y tampoco te deja, pues lo haces con "-v " al final. Al hacer "vi /etc/ssh/sshd_config" vemos que hemos modificado el puerto pero que tenemos la línea comentada por defecto. Lo descomentamos y escape :wq y reiniciamos el sistema. Dara un fallo y ponemos "systemctl status sshd.service" y no ve nada. Y hace "journalctl -xe" ve que intentamos enlazar al puerto 22322 y da un fallo de permisos. Para terminar y arreglarlo vamos a "vi /etc/ssh/sshd_config" te explica que debemos avisar al so de que vamos a cambiar el puerto con:

```
SELinux about this change.
semanage port -a -t ssh_port_t -p tcp #PORTNUMBER
```

UBUNTU:

- Atp-get -> en desuso
- Apt es equivalente a yum
- Para buscar un proceso perdido:
 - Ps | grep ssh
 - No muestra nada porque ninguno cuelga de este bash
 - Ps -Af | grep ssh
 - Ahí si aparece
- Vamos a instalar ssh en Ubuntu:
 - Sudo apt instal tasksel
 - Sudo tasksel
 - Damos en openssh y OK
 - Apt search ssh | grep server
 - Sudo Apt install openssh-server
 - Es la alternativa a lo de tasksel y así sabemos que versión instalamos
 - Ps -Af | grep ssh
 - Vemos que está en ejecución y eso esta mal, porque el tasksel no tiene permisos para arrancar el permiso y aun así lo pone.
- Para configurar lo de root:
 - Vi /etc/ssh/sshd_config
 - Vemos que esta prohibido root con contraseña por defecto

LECCION 2

COPIAS DE SEGURIDAD Y CONTROL DE VERSIONES (CON GIT)

Pregunta de examen: ¿Opciones mas comunes de resync?

COPIAS VS BACKUP

Si haces Ctrl C de un archivo, has hecho una copia, pero la diferencia de un backup es que este tiene una metainformación o metadatos superiores sobre versiones, fechas, etc... El backup también se puede colocar físicamente en otro sitio. El backup es una ciencia para él.

Comandos para las copias de seguridad:

Dd -> copia binaria bit a bit y se puede utilizar a día de hoy para recuperar un disco o un dispositivo móvil. Tenemos un inputfile y un outputfile, creamos nuestra copia de archivos, podemos especificar un archivo de imagen y para deshacer el cambio, cambiamos el orden de los archivos y así nos lo llevamos a nuestro disco. La entrada de bloque está a 512 bytes y podemos poner 4Kb para que sea más rápida. Si haces eso, para recuperar el contenido de una tarjeta después dejas quieta la tarjeta y jugamos con el archivo generado o el dispositivo donde hayamos copiado la imagen de la información original, así, de tener que deshacer alguna modificación, aun se puede. El problema de dd es que nos permite hacer copias pero la parte de metadatos tenemos que ponerla nosotros (añadiendo al nombre de archivo la ejecución, ubicación, etc) Como ventaja es que tenemos la copia exacta bit a bit

Cpio-> Coge el listado de archivos de la entrada estándar y lo convierte en un único archivo (pero no comprime). Aunque está en desuso, las ventajas de esto son que si usamos sftp para llevarte las copias de seguridad a otro sitio y queremos enviar millones de archivos, puede ser conflictivo y crea un tráfico innecesario en la red, entonces con cpio coge esos millones de archivos, lo mete en uno solo y se ahorra tiempo. Otra cuestión es que si nos preocupa el tiempo que tarde en enviarse los archivos. Siempre queremos que sea cuanto más rápido, pero además, en el caso de que se interrumpa la copia. Para esto, tenemos en cuenta la frecuencia del backup (por ejemplo algunas cosas las quiero cada 12h, o 24h o algo así) entonces si tardo en hacer la copia más que lo que hay entre copia y copia, no las estoy haciendo bien. Una posible solución es compresión pero el tío se refiere a una cache. Poner un servidor intermedio al que mandar los datos muy rápido y ya lleva la información a ese tercer lugar, que es el destino original el tiempo que necesite. El intermediario en modo Bach en cola va llevando la información a la ubicación tercera. Es como un proxy o buffer. Otra posible solución es que no necesitamos cada X tiempo el 100% de los datos, sino que solo necesitamos guardar lo que hemos modificado. Eso es un backup incremental: tenemos una imagen base y vamos almacenando las diferencias y es todo lo que mandamos. Pasa de "26" horas a "1" h. También nos ahorramos espacio. Otra cosa a tener en cuenta, la información delicada. Si va a tener información delicada la nube a la que mandamos la copia, mejor que esté en España o

Europa para que este vigente la LOPD (española) y GDPR (europa) y ya como mucho mirar convenios para países asociados como canada (nunca china, ni EEUU, ni Australia, ni Mexico).

Tar-> sustituo de cpio. La única diferencia es que en vez de tener que especificar en la entrada estándar el listado de archivos, se le especifica un conjunto de archivos o directorios que empaquetara. También tiene la opción de comprimirlo y de ahí la extensión .gz. En la orden "tar cvzf MyImages..." cvzf significa create overboss zip file, que es en modo overboss y comprimido. Para comprimirlo es "tar -xvf public_html.tar" que es x-> extract v->overboss f->file. SFTP lleva la S delante para que sea seguro o no estaremos respetando la LOPD. Sabernos el tar de memoria.

- **cpio**
 - Copia archivos a y desde (archivos)
 - `ls | cpio -ov > /tmp/object.cpio`
 - `cpio -idv < /tmp/object.cpio`
- **tar**
 - Ejemplos:
 - `tar cvzf MyImages-14-09-17.tar.gz /home/MyImages`
 - `tar -xvf public_html-14-09-17.tar`

Rsync -> Dropbox de los viejos. Hacían sincronizaciones a mano. Permite sincronizar una fuente y un destino de los datos y se puede hacer a través de ssh. Así se cumple la normativa legal. Además permite los envíos incrementales y reduce el tiempo que necesita. Puede agrupar todos los cambios en un archivo único y si sincronizamos dos directorios y hemos borrado algo sin querer, tenemos un problema porque lo borra.

Otra cosa importante en los backups es el tiempo de recuperación. Este hombre dejaba var en var_OLD porque si queremos volver a la configuración original, con hacer un `mv /var_old /var`, estaba todo listo. Hay que tener cuidado con --delete en rsync porque borra. Se prueba:

Crear un archivo: `mkdir 1 mkdir2 touch 1` y cosas así y probar a hacer `rsync, --delete` y cosas así. Hay que probarlas para saber usar el `. * / username, maquina` y eso. Para usar ssh para que vaya cifrado es `" -e "ssh" "`. Hay que saber de memoria el uso común de rsync:

`rsync -aviz /home usu@maquina:/backup`

La iz del final es que haga un informe y que vaya comprimido. Para restaurar con rsync invertimos el orden del origen y del destino. Lo usaremos con algún cluster. Va a preguntar si o si las opciones más comunes, precauciones a tomar, como hacer transferencia de información, etc

SO para backups-> tartarus, backup2l, duplicity, sftpclone que implican un servicio completo con BD, demonios y cosas así. Es más compleja y tiene sobrecargar mayor en el sistema. A veces puede interesar monitorizar el estado del backup sin ir a la consola. Error para que aprendamos:

Todo esto puede pasarle fácilmente...

Destroyed Working Backups with Tar and Rsync (personal backups)

I had only one backup copy of my QT project and I just wanted to get a directory called functions. I end up deleting entire backup (note -c switch instead of -x):

```
cd /mnt/backupusbharddisk  
tar -zcvf project.tar.gz functions
```

I had no backup. Similarly I end up running rsync command and deleted all new files by overwriting files from backup set (now I have switched to [rsnapshot](#))

```
rsync -av -delete /dest /src
```

Again, I had no backup.

CONTROL DE CAMBIOS

Se puede hacer un cp pero es cutre porque no se añaden metadatos. No es lo ideal aunque pueda ser útil. GIT es un sistema de control de versiones. Hay herramientas que permiten guardar etc por ejemplo con git. Ventajas:

- **Ventajas (para ISE)**
 - Ponemos un pie en devops
 - **Seguimiento de los cambios (y de su autor)**
 - Permite planificar un entorno de prueba entre varias personas
 - ...

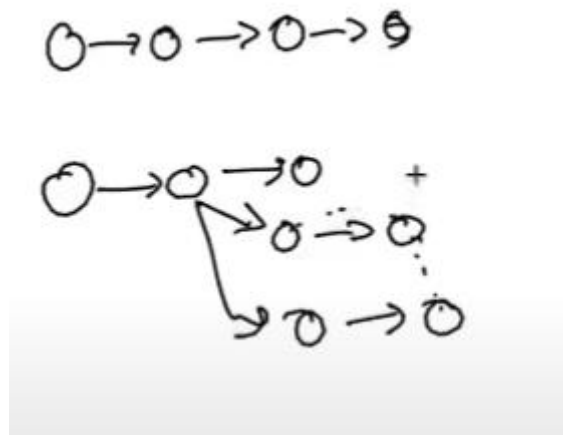
Desarrollo git -> Linus Benedict Torvalds. Github ahora pertenece a Microsoft. Desarrollo el kernel de Linux y alternativamente git. Github establece un frontend web para sus servidores que están ejecutando los servicios de git.

Cómo funciona git:

Tenemos un directorio de trabajo, trabajamos con estos archivos y los cambiamos, por lo que los marcamos para seguimiento y lo metemos a nivel abstracto en una zona que es el stage y en un paso final se pasa al repositorio. Ahí tenemos la instantánea actual de nuestro proyecto que puede ser código o cualquier otro tipo de archivos. No aportaría mucho que lo hiciésemos sobre imágenes porque se realiza una compresión que no podríamos hacer con imágenes y se nos iría el espacio. Mejor para archivos de texto. Cuando hacemos un commit, que es pasar de los archivos de seguimiento (los cambiados y marcados para seguir, que por ejemplo contraseñas y tal no se marcan para seguir) al repositorio. Cuando hacemos esos commit, se genera un hash de unos 40 caracteres para identificar esa transacción aunque con los 7 primeros suele valer. Esos commit se van colocando en forma de grafo dirigido acíclico y ese ultimo commit es un puntero denominado a head. Podemos hacer referencias relativas de movernos hacia atrás x posiciones y cosas así:

- Podemos desplazarnos a partir del identificador (con ~ y ^):
 - HEAD~3 == HEAD^^ == HEAD^3
 - Se pueden combinar: HEAD~3^2
- Podemos desplazarnos a partir del identificador (con ~ y ^):
 - HEAD~3 == HEAD^^ == HEAD^3
 - Se pueden combinar: HEAD~3^2

si nos perdemos, nos reubicamos con git reflog. Cuando veamos la salida de git veremos los dos puntos que significan un intervalo lineal entre un commit y otro, y si son 3 puntos es que no es lineal. Lo de lineal significa que vamos haciendo commits y puede que se ramifiquen, entonces de algún commit a otro no haya referencia lineal:



Tenemos unos blobs (binary large objects) que son archivos en binario. Entonces tiene los datos y los inodos correspondiente. Cada objeto tiene su hash identificativo, por lo que git es un sistema de archivos que es direccionable por contenido, ya que este esta contenido dentro del blobs. Al final lo que tenemos son llaves y valores identificativos. Se pueden poner etiquetas también a los objetos.

AHORA VA A LA TERMINAL DE CENTOS PARA HACER PRUEBAS:

WORKFLOW: servidor central donde tenemos un repositorio con dos equipos de trabajo y cada uno tiene su directorio de trabajo, su zona de seguimiento y su repositorio propio. Y un servidor que va a ser centos. No viene instalado por defecto:

- Dnf/yum install git
- Git init -> iniciar un repositorio
- Crear un directorio para los repositorios:
 - o Sudo mkdir /git
 - Practicas,ISE
- Sudo dnf install git
 - o si
- sudo dhclient
 - o para lanzar el cliente de dhcb o dhcp o algo asi para que refresque la ip de np0s3 y np0s8.
- Cd /git
- Si haces git init te da fallo porque como lo has creado en root entonces el propietario es root y el grupo es root (dar permiso de 777 es mejor que no, es una locura)
- Creamos un usuario git y añadimos el resto de usuarios al grupo de git.
 - o Sudo adduser git
 - o Groupadd le da fallo asi que prueba otra cosa
 - o Sudo vi /etc/group
 - Abajo del todo junto a "git:x:...." escribe: Alberto:Ana
 - :wq
 - o Sudo chgrp -R git /git
- Primero el añadimos el 77 en los permisos
 - o Sudo chown -R git /git
 - o Ls -la / -> para probarlo
 - o Sudo chmod 770 /git
 - o Ls -la / -> comprueba
- Git init repo1 - -bare
 - o Le da fallo y se tira la vida. Termina haciendo trampas.
 - o Lo que hemos hecho con el init es iniciar el bare repo. El bare no tiene directorio de trabajo, con respecto a los repos normales.
 - o Ls -> se ve repo1
 - o Cd repo1
 - o Ls -la -> se vera:

```

alberto@localhost repo1$ ls
branches config description HEAD hooks info objects refs
alberto@localhost repo1$ ls -la
total 16
drwxr-xr-x. 2 alberto alberto 119 oct 29 12:03 .
drwxr-xr-x. 3 git      git      19 oct 29 12:03 ..
drwxr-xr-x. 2 alberto alberto  6 oct 29 12:03 branches
-rw-rw-r--. 1 alberto alberto 66 oct 29 12:03 config
-rw-rw-r--. 1 alberto alberto 73 oct 29 12:03 description
-rw-rw-r--. 1 alberto alberto 23 oct 29 12:03 HEAD
drwxr-xr-x. 2 alberto alberto 4096 oct 29 12:03 hooks
drwxr-xr-x. 2 alberto alberto 21 oct 29 12:03 info
drwxr-xr-x. 4 alberto alberto 30 oct 29 12:03 objects
drwxr-xr-x. 4 alberto alberto 31 oct 29 12:03 refs

```

- Sudo useradd Ana
 - o Practicas,ISE
- Sudo vi /etc/ssh/sshd_config
 - o Pone el port 22
- Systemctl start ssh

- Abre una terminal y se mete en el usuario Ana y en otra terminal esta dentro de Alberto. Ahora vamos a traernos el repositorio del servidor desde alberto
 - o Git clone IP:/git/repo1
 - Se obtiene la IP con "Ip addr" y mira la 3
 - Practicas,ISE
- Ana se va a su home y hace:
 - o Git clone IP:/git/repo1
 - Practicas,ISE
- Alberto entra dentro de repo1
 - o Cd repo1/
 - o Ls -la

```
alberto@localhost:~/BORRAME/repo1$ ls -la
total 12
drwxr-xr-x. 3 alberto alberto 4096 oct 29 17:08 .
drwxr-xr-x. 4 alberto alberto 4096 oct 29 17:08 ..
drwxr-xr-x. 7 alberto alberto 4096 oct 29 17:08 .git
```

- Ana hace lo mismo.

```
(Ana@localhost ~)$ cd repo1/
(Ana@localhost repo1)$ ls -la
total 12
drwxrwxr-x. 3 Ana Ana 4096 oct 29 17:08 .
drwx----- 5 Ana Ana 4096 oct 29 17:08 ..
drwxrwxr-x. 7 Ana Ana 4096 oct 29 17:09 .git
```

- Lo que vemos es el working directory. Lo particular es el subdirectorio de .git. Es lo que tenemos en el bare repo. Si tengo nose cuantos commit y se te escapa un rm repo1, la información se pierde porque es donde esta la info. De las 3 copias, las 3 son validas. Pueden tener sus cambios locales pero a nivel de repositorio son validos.
- Alberto se va a repo1:
 - o Touch README.md
 - o Git status -> mucha información como que readme es archivo sin seguimiento. Que si queremos seguirlo es con git add <nombre de archivo>
 - o Git commit -> para actualizar las repos, pero no hay nada
 - o Git add README.md
 - o Git commit -m "hemos creado el archivo README.md"
 - o Ls .git/ -la -> hemos cambiado el head
 - o Git status -> no hay nada porque esta todo actualizado. Hemos actualizado el repo de Alberto pero no el bare y lo hacemos con push y pull
 - o Git push
 - Practicas,ISE
- CENTOS:
 - o En repo1 seguimos viendo lo mismo, por que? Porque es tipo bare y no hay working directory
- Ana:
 - o Ls -> dentro de repo1 y no hay nada
 - o Git pull
 - Practicas,ISE
 - o Ls -> ya se ve README.md
- Alberto:

- Vi README.md -> dentro escribe # Empezamos con git...
- Git status
- Git diff -> vemos las diferencias entre el working directory y el área de seguimiento.
- Git add README.md
- Git diff -> no muestra nada porque ya no hay diferencia entre lo del working dir y el área de seguimiento
- Git diff HEAD -> diferencia entre wd y el repositorio y se ve que hemos añadido esa línea.
- Git commit -m "Actualizamos README.md"
- Git diff -> nada
- Git diff HEAD -> nada
- Git diff - -staged -> nada
- Vi README.md
 - Añade "#Titulo:"
- Git diff -> salen cosas
- Git diff HEAD -> también
- Git add README.md
- Git diff -> ya no sale nada
- Git diff HEAD -> sale la diferencia
- Git diff -staged -> diferencia entre lo del repo y el seguimiento
- Git commit -m "Añadimos titulo"
- Git push
- Ana:
 - Git fetch
 - Git merge
 - Cat README.md
- Alberto
 - Vi README.md
 - Pone después de titulo: en un lugar...
 - Git commit -a -m "readme con titulo"
- Ana:
 - Vi README.md
 - Pone después de titulo: otro titulo
 - Git add README.md
 - Git commit -m "titulo nuevo"
 - Git push -> le da fallo

COSAS IMPORTANTES A TENER EN CUENTA:

-m "mensaje" en los commit, el mensaje tiene que ser ilustrativo y significativo. Los commits deben ser lo mas atomicos posibles. A lo mejor no cada vez que cambiemos una línea pero si hacerlos atomicos porque nos permitirá luego encontrar esa aguja en el pajar. Se busca el mensaje y lo encontramos rápido. Si no, haciendo un commit con 100 lineas, habría que revisarlas todas. Lo ideal es que los commit, cuanto mas pequeños mejor. Aporta trazabilidad.

Pull lo que hace es traer información del servidor y las une. (fetch y merge). Eso es que te la descargas del servidor y luego la recuperas. Git pull = git fetch + git merge. La cosa es que si

una persona hace un commit y otra hace otro commit y se adelanta, le va a dar fallo y tendrás que hacer el push a mano.

LECCION 3

SSH viene instalado en Centos pero en Ubuntu server no, nos pregunta en la instalación si lo queremos. En cuanto a nomenclatura, en centos es sshd y en Ubuntu es ssh y sshd. Sobre root puede entrar en Ubuntu no con password y en centos el permit root login esta a yes. El firewall esta activado por defecto en cetos (firewallcmd) y en Ubuntu no, aunque lo tiene (ufw), pero desactivado.

Sobre un fallo que le dio la semana pasada: Existen unos bits que hacen que un proceso pase a ejecutarse en modo superusuario, entonces con un chmod 2770 seria suficiente para que le deje hacer cosas con git. Entonces por la tarde, al no hacerlo como root, esta desactivado por defecto y no tenia permisos.

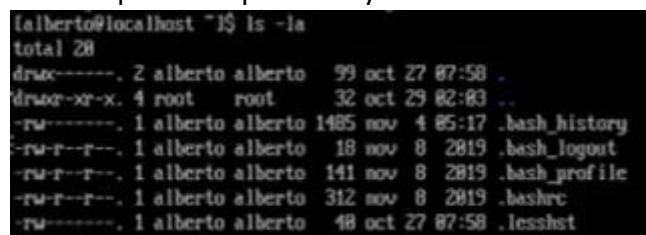
PARA CAMBIAR EL PUERTO:

Hay que modificar: /etc/ssh/sshd_config

Había una #Port 22 y hay que editar con “sed” o da fallo. Hay que quitar la almohadilla. Una vez hecho el cambio de puerto vemos que ocurre en centos y Ubuntu.

Ssh-keygen lo que hace es que si en el cliente ejecutamos esa orden, se generan una llave publica y una privada. Se utiliza un intercambio de clave para cifrar la comunicación. En el protocolo los servidores mandan la información cifrando con la publica y solo se puede descifrar con la privada que solo conocemos nosotros. Es un cifrado asimetrico por haber dos llaves diferentes.

- ALBERTO:
 - Su -> pasa a super usuario para poder modificar sshd
 - Vi /etc/ssh/sshd_config
 - Quita # del port y cambia de 22 a 22022
 - Systemctl restart sshd
 - Ssh IP -p 22022 -> para comprobarlo
 - Logout -> se sale
- SE PASA A CENTOS PORQUE TENIA YA EL PAR DE CLAVES GENERADAS:
 - Ls -la -> para ver que tiene y le sale



```
alberto@localhost ~$ ls -la
total 28
drwx----- 2 alberto alberto  99 oct 27 07:58 .
drwxr-xr-x  4 root    root    32 oct 29 02:03 ..
-rw-----  1 alberto alberto 1485 nov  4 05:17 .bash_history
-rw-r--r--  1 alberto alberto  18 nov  8 2019 .bash_logout
-rw-r--r--  1 alberto alberto 141 nov  8 2019 .bash_profile
-rw-r--r--  1 alberto alberto 312 nov  8 2019 .bashrc
-rw-----  1 alberto alberto  48 oct 27 07:58 .lesshst
```

- Ssh-keygen -> nos pregunta si queremos ubicar la llave en algún lugar por defecto. Le puedes dar a enter pero ojo, es importante quien tenga

esa llave. Por defecto dentro de home y un directorio oculto esta bien. Luego te pregunta una contraseña por si alguien intenta entrar a ese archivo. Lo deja en blanco dos veces y genera la llave privada

- Ls -la -> vemos el directorio .ssh
- Ls -la .ssh/ -> y vemos las dos llaves, la privada y la publica. En los permisos se nota.
- Scp -> no lo hace, es algo que se podría hacer para distribuir la publica
- Ssh-copy-id -> hace el procedimiento de forma automática y transparente y podemos especificar el puerto
- Ssh-copy-id IP -p 22022 -> le da fallo
- Sudo dhclient -> practicas,ISE
- Ssh-copy-id IP -p 22022 -> pide la contraseña del usuario que es practicas,ISE
- Ssh IP -p 22022 -> para logearse y le deja, con eso entra en Ubuntu. Asi no hay que teclear la contraseña al conectarnos al servidor. Util si hay 1000 nodos y vamos a pasar mensajes. Nos vale para todos los servidores a los que nos vayamos a conectar. Si la perdemos tenemos que crear ambas de nuevo pero tenemos que borrar la publica del servidor o ese usuario si tendrá acceso.
- UBUNTU:
 - Vamos a limitar el acceso por contraseña:
 - Sudo vi /etc/ssh/sshd_config
 - Passwordauthentication no -> es lo que tenemos que descomentar y poner, pues esta a yes por defecto
 - Systemctl restart sshd
 - Practicas,ISE
- CENTOS:
 - Ssh IP -p 22022
 - Puede hacerlo perfectamente, pero si lo hace desde su anfitrión (misma orden) no le deja.
 - Ssh-copy-id IP -p 22022 -> para copiar las claves. No te va a dejar. Para que te deje se va a Ubuntu de nuevo
- UBUNTU:
 - Sudo vi /etc/ssh/sshd_config
 - Pone a yes el passwordauthentication por un tiempo limitado. Se puede hacer con el "sed"
 - Systemctl restart sshd
- DESDE SU ANFITRION:
 - Ssh-copy-id IP -p 22022 -> Ya si le deja
- UBUNTU:
 - Sudo vi /etc/ssh/sshd_config
 - Pone a no el passwordauthentication
 - Systemctl restart sshd
- DESDE SU ANFITRION:

- Ssh IP -p 22022 -> dentro de Ubuntu sin teclear contraseña
- logout
- CENTOS:
 - Su
 - Vi etc/ssh/sshd_config
 - Ponemos el puerto 22022 y descomentado. Para modificar esto hay que comentárselo a SELinux y al meter "semanage" no lo tiene
 - Dnf provides semanage -> ve el paquete que tiene semanage que es policycoreutils o algo así
 - Dnf install policycoreutils-python-utils
 - S
 - Semanage port -l | grep ssh -> enseña los puertos que está usando ssh que es el 22
 - Semanage port -a -t ssh_port_t -p tcp 22022 -> vamos a manejar un puerto y la a es de añadir. Es normal que tarde unos segundos.
 - Semanage port -l | grep ssh -> vemos el cambio. Ya no va a dar fallo el cambiar el puerto.
 - Systemctl restart sshd
 - Systemctl status sshd -> está en el 22022
- SU HOST:
 - Ssh IP -p 22022 -v -> da fallo porque no hay ruta al host. Esto no ha pasado con Ubuntu.
- CENTOS
 - ssh localhost -p 22022 -> para ver si das servicio a tu propia máquina y veo que sí puedo.
 - Logout
 - ¿Entonces que hay que me impide el acceso desde fuera? Un cortafuegos. Vamos a modificarlo
- UBUNTU:
 - ufw status -> se ve desactivado
 - ufw enable -> ya lo tenemos
- SU HOST:
 - ssh IP -p 22022 -> ya no te deja
- UBUNTU:
 - ufw allow 22022
- SU HOST:
 - ssh IP -p 22022 -> ya te deja
- CENTOS:

```
firewall-cmd --add-port=443/tcp
firewall-cmd --permanent --add-port=443/tcp
```

- Con la primera línea se cierra al reiniciar, la segunda es permanente.
- Firewall-cmd --permanent --add-port=22022/tcp

- HOST:
 - o Ssh IP -p 22022 -> sigue sin acceso porque hemos añadido el puerto pero hay que recargar el cortafuegos o poner la primera línea para que en esta ejecución se tenga en cuenta.
- CENTOS:
 - o Firewall-cmd --add-port=22022/tcp
- HOST:
 - o Ssh IP -p 22022 -> ya si

Para acceder sin contraseñas haces lo del ssh-copy-id IP -p 22022 y ya se emparejan las claves y no te la pide más. Podemos poner ya todos los passwordauthentication a no.

Mirar en el guion screen, terminator y tmux (evolución de screen).

Fail2ban -> si fallas en el acceso vas a la lista de baneados. Si alguien se equivoca un numero determinado de veces, cancelan tu IP.

- CENTOS:
 - o Su dnf install epel-release
 - o Dnf search fail2ban
 - o Dnf install fail2ban
 - o Systemctl status fail2ban -> cargado pero inactivo
 - o Systemctl enable fail2ban -> en el próximo reinicio arrancara, pero esta inactivo aun
 - o Systemctl start fail2ban -> activo y habilitado
 - o Fail2ban-client status -> hay 0 carceles definidas
 - o Cd /etc/fail2ban
 - o Ls -la:

```
[root@localhost alberto]# cd /etc/fail2ban/
[root@localhost fail2ban]# ls -la
total 64
drwxr-xr-x. 6 root root 158 nov 5 10:25 .
drwxr-xr-x. 83 root root 8192 nov 5 10:25 ..
drwxr-xr-x. 2 root root 4096 nov 5 10:25 action.d
-rw-r--r--. 1 root root 2817 ago 28 07:55 fail2ban.conf
drwxr-xr-x. 2 root root 6 ago 28 07:55 fail2ban.d
drwxr-xr-x. 3 root root 4096 nov 5 10:25 filter.d
-rw-r--r--. 1 root root 25740 ago 28 07:55 jail.conf
drwxr-xr-x. 2 root root 31 nov 5 10:25 jail.d
-rw-r--r--. 1 root root 2827 ago 28 07:55 paths-common.conf
-rw-r--r--. 1 root root 930 ago 28 07:55 paths-fedora.conf
[root@localhost fail2ban]#
```

- o Less jail.conf -> nos dice que no modifiquemos el archivo. Lo vamos a copiar en el local para conservar los fallos si algo se restaura.
- o Cp -a jail.conf jail.local
- o Vi jail.local -> vamos a sshd y en enabled ponemos true
- o :wq
- o Vi /etc/ssh/sshd_config -> temporalmente ponemos a yes el passwordauthentication

- Systemctl restart sshd
- UBUNTU:
 - Ssh IP -p 22022 -> fallamos la contraseña unas cuantas de veces
- CENTOS:
 - Systemctl restart jail2ban.service
 - Fail2ban-client status -> vemos 1 ya y si desde Ubuntu intentamos entrar, no podemos
 - Fail2ban-client status sshd -> vemos la ip baneada.
- UBUNTU:
 - Ssh IP -p 22022 -> nos sigue dejando
- CENTOS:
 - vi jail.local -> tiene un parámetro para el puerto que vamos a cambiar porque por defecto esta el 22. Buscamos la parte de sshd y en port ponemos 22022
 - systemctl restart fail2ban.service
 - fail2ban-client status sshd
 - iptables -L -> rechaza todos los paquetes a Ubuntu server y con este comando se ve.

Rkhunter-> es un software malicioso que intenta esconderse (no es un virus por la replicacion) pero puede filtrar contraseñas. Con RootKit Hunter, semanalmente, podemos estar seguros de ver que no hay rkhunter.

AHORA VAMOS A CONFIGURAR EL SERVICIO LAMP

HTTP -> Hyper Text Transfer Protocol

HTML -> Hyper Text MARK-UP Language

Los hipertextos se escriben usando el lenguaje de markups html. En el tenemos el texto, el texto especial entre angulo para marcar en negrita, un enlace, titulo, etc.

Tenemos un servidor escuchando peticiones en el puerto 80, le llega una petición y quiere un documento html, el servidor lo devuelve pero es que los html son estáticos y no se pueden cambiar. Así que si quisiéramos tener un documento con comentarios de los servidores podríamos tener el formulario como tal pero no podría subirse al servidor a no ser que tengamos un esquema mas complejo. Este dinamismo se añade con una lógica que compruebe si el usuario puede escribir en el documento y una base de datos. Por lo que tendremos una lógica y una BD. Podemos meter lógicas en el html con script y usaríamos javascript, pero este se ejecuta en el cliente. Por temas de seguridad con las contraseñas, es obligatorio una lógica en el servidor y un esquema que me permita almacenar datos. Html no tiene estado. Conocemos el sistema de gestión de bases de datos MySQL y MariaDB. Si quieres montar una web, que conjunto de programas podremos tener: un servidor con el puerto 80 que devuelva html

(apache o nginx por ejemplo) y te sirven PHP, Python como lógicas y MySQL como BD. Si juntas todo en una arquitectura Linux, tenemos la pila software LAMP (WAMP o XAMP para mac).

- UBUNTU:

- Sudo taskset
 - Marcamos LAMP server
- Man curl -> para comprobar
- Curl localhost
- Mete la IP dentro del buscador a ver pero se cuelga. Es porque el cortafuegos esta activo
- Sudo ufw status -> lo vemos
- Sudo ufw allow 80 -> ahora si, vamos de nuevo al buscador y nos debería Salir ya la pagina de apache
- Php -a -> para ver que esta funcionando
- Sudo mysql -> funcionando tambien

- CENTOS (misma instalación a mano):

- Dnf search apache -> vemos:

```
===== Coincidencia en Resumen: apache =====
httpd.x86_64 : Apache HTTP Server
```

- Dnf install httpd
 - S
- Systemctl status httpd -> inactivo y disabled
- Systemctl enable httpd -> activao papaaaa
- Systemct start httpd
- Systemctl status httpd
- Curl localhost -> nos devuelve la pagina
- Dnf search php :

```
===== Coincidencia en Nombre , Resumen: php =====
php.x86_64 : PHP scripting language for creating dynamic web sites
```

- Dnf install php
 - S
- Dnf search mysql

```
===== Coincidencia en Nombre , Resumen: mysql =====
mysql.x86_64 : MySQL client programs and shared libraries
mysql-zem-backup : MySQL backup manager
```

- Dnf install mariadb-server
 - S
- Systemctl status mariadb -> inactivo y disabled
- Systemctl enable mariadb
- Systemct start mariadb
- Mysql -> accedemos bien
- Mysql_secure_installation

- Te pide contraseña pero es ninguna, peligroso. Le ponemos practicas,ISE a root. Tambien tenemos un usuario anonimo pero no nos interesa y lo quitamos. No queremos que root se conecte desde una maquina de fuera. Borramos también la base de datos de test. Si cargamos la tabla de privilegios.
- Mysql -> ya no nos deja
- Mysql -u root -p
 - Practicas,ISE
- Firewall-cmd --add-service=http --permanent
- Firewall-cmd --reload -> ya podemos desde el navegador meter la ip
- Dnf install php-mysqld.x86_64
 - Si

EJEMPLO DE PAGINA:

SU HOST:

- Touch /var/www/html/miscript.php
- Vi /var/www/html/miscript.php
 - Ctrl + shit + v
 - Modificamos "my_user" = root, "my_password" = practicas,ISE "my_db" = db
- Curl localhost
- Mete en un buscador la IP
- Firewall-cmd --add-service=http --permanent
- Firewall-cmd --reload
- Vuelve a meter la IP y ya si sale
- Busca en el buscador: IP/miscript.php -> no sale bien
- Vi /etc/httpd/conf/httpd.conf
 - Buscamos directoryindex y lo ponemos asi

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

- :wq
- Systemctl restart httpd
- Vuelve al buscador y sigue dando fallo. Mira los fallos dando a f12 y dando a consola. Le pone http500 es un fallo del servidor.
- Para ver el error "less /var/log/httpd/error_log" y no ve nada, se va a "less /var/log/httpd/access_log" y vemos los fallos 500. Ahora va a "less /var/log/php-fpm/www-error.log"
- Dnf install php-mysqld.x86_64

- S
- Otra forma de comprobar el error: cd /var/www/html y luego php miscript.php. Nos metemos en mysql -u root -p y ponemos CREATE DATABASE db;
- Vuelve al buscador, y ooooooooootro fallo.
- Less /var/log/Audit/Audit.log -> pone que no da permiso a php-fpm
- Getsebool -a -> tenemos booleanos que establecen políticas y restricciones.
- Getsebool -a | grep http -> y hay uno que es httpd_can_network_connect_db
- Setsebool httpd_can_network_connect_db=1
- FUNCIONA SI TE VAS AL BUSCADOR

Cogeremos este texto:

```
<?php
$enlace = mysqli_connect("127.0.0.1", "mi_usuario", "mi_contraseña"
, "mi_bd");

if (!$enlace) {
    echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
    echo "errno de depuración: " . mysqli_connect_errno() . PHP_EOL
;
    echo "error de depuración: " . mysqli_connect_error() . PHP_EOL
;
    exit;
}

echo "Éxito: Se realizó una conexión apropiada a MySQL! La base de
datos mi_bd es genial." . PHP_EOL;
echo "Información del host: " . mysqli_get_host_info($enlace) . PHP
_EOL;

mysqli_close($enlace);
?>
```

SCREEN, TERMINATOR Y TMUX

Terminator -> permite varias configuraciones, ponerlo mas grande, diferentes colores y es útil para cuando trabajamos con varios servidores. Eso mismo en modo texto seria tmux, que es una evolución de screen. En ssh hay un parámetro que es keepalive pero tras mucha inactividad, la conexión se cierra. Si se queda a medio hacer, al hacer un "ps xf" vemos la rama del proceso sshd que entra por un bash y tal, el resto de hijos, en cascada, al cerrar la conexión y terminar el proceso de golpe, se cerraran también. Se puede evitar poniendo screen o tmux para crear un proceso que adoptara a los procesos hijo y se quedan colgados del tmux que queda residente.

Sudo dnf install screen -> para instalarlo. Ahora hace un vi de lo que sea, por ejemplo, vi asdf y lo deja en background, poniendo ps xf vemos que cuelga vi también entre los procesos. Entonces hacemos screen y luego ps xf para ver que si que cuelga de screen ya. Ahora vi firfjif y lo dejas en background, hacemos ps xf y vi y ps cuelgan de screen. El anterior vi si que cuelga de bash. Entonces si muere el proceso, si hace screen -r (detrás de la orden se puede especificar)y se puede listar las sesiones con screen -list