SSH

Servidor:

A sistemes Debian/Ubuntu/Fedora/CentOS cal instal.lar el servidor SSH, disponible mitjançant el paquet "openssh-server". Un cop instal.lat, a Debian/Ubuntu es podrà iniciar/atura/habilitar/deshabilitar així: *systemctl {start | stop | enable | disable } ssh* i a Fedora/CentOS així: *systemctl {start | stop | enable | disable } sshd*

L'arxiu de configuració del servidor és /etc/ssh/sshd_config . Les directives més importants que hi podem trobar són:

Port nºport : port per on escoltarà peticions
ListenAddress una.IP : permet restringir l'escolta només a una determinada tarja. Si es posa la IP 0.0.0.0 vol dir "totes les interfícies"

Compression yes/delayed/no: autoritza a què es puguin compactar les dades a transmetre ("delayed" és yes però només després d'autenticar) si així ho demana el client.

Ciphers algo1,algo2,...: indica els algoritmes d'encriptació a usar (per ordre de preferència) durant la comunicació. Algun ha de coincidir amb els indicats al client per tal de què la comunicació es pugui realitzar.

X11Forwarding yes : autoritza que els clients puguin fer servir el paràmetre -X
X11DisplayOffset 10 : especifica el display del client on es veuran les aplicacions gràfiques "forwardades" (per no interferir amb servidors X reals).

PasswordAuthentication yes/no : ofereix la possibilitat d'utilitzar aquest mètode d'autenticació
PermitEmptyPasswords yes/no : obvi
PubKeyAuthentication yes|no : ofereix la possibilitat d'utilitzar aquest mètode d'autenticació
PermitRootLogin yes/no/without-password : el valor "without-password" vol dir que només s'autoritza el root si s'usa un sistema d'autenticació que no faci servir contrasenyes (com ara el PubKey, per exemple)

AllowUsers usuari??@192.168.0.2?? unaltre: els usuaris que no hi apareixen, tindran denegat l'accés
DenyUsers usuari??[@192.168.0.*] unaltre none : els usuaris que no hi apareixen, tindran permís per entrar
També està AllowGroups i DenyGroups. A més de * i ?, es pot utilitzar ! per negacions

Match ... : Les directives Match poden escriure's vàries vegades. Serveixen per marcar que les directives posteriors (per exemple Banner, ForceCommand, etc) només s'aplicaran pels casos concrets que coincideixin amb la condició del Match (usuari,nom de màquina ó ip), fins arribar al final del fitxer o bé a una altra directiva Match. Exemples:

```
Match User usuari??,usuario*,!usuari3
Match Host maq??,maq*,!maquina3
Match Address ip/mask, !ip/mask
```

MaxSessions nº : número de conexions simultànies permeses
MaxAuthTries nº :número de reintents que es permeten per intentar escriure bé la contrasenya
MaxStartups nº: número de conexions simultànies permès que no han acabat d'autenticar-se
LoginGraceTime nºsec : temps per introduir les credencials (0 no hi ha límit)
ClientAliveInterval nºsec : temps que s'espera sense rebre dades per tancar la conexió definitively
ClientAliveCountMax nºmissatgesqueservidorenviaenarribaralClientAliveInterval(1cadacop)abansdedesconectar

AcceptEnv var1 var2 ... : (veure directiva SendEnv del client)

ForceCommand /ruta/comanda/que/seexutara/al/servidor/un/cop/loguejat: Sobreescriu la possible comanda que es pugui haver escrit al client -la qual es guardarà a la variable SSH_ORIGINAL_COMMAND per si de cas-

PrintMotd yes/no : si val “yes” mostrarà el contingut de /etc/motd un cop loguejat
Banner /ruta/arxiu/amb/missatge/que/es/mostrara/abans/del/login

Subsystem sftp /ruta/servidor/sftp : activa la possibilitat de funcionar com servidor Sftp

Client SSH (accés remot):

La forma general d'executar el client és:
NOTA:Als exemples “user1” serà un usuari existent (i vàlid) a la màquina client i "user2" ho serà a la màquina servidora.

ssh [user2@]servidor [comanda]

Si s'escriu la comanda, no s'obre cap terminal: s'executa la comanda remotament i ja està. Si no s'escriu l'usuari, s'entrarà al servidor amb el mateix usuari, si existeix. Una altra manera d'executar el mateix seria: *ssh -l user2 servidor [comanda]*

Paràmetres que es poden afegir:

-p nºport (si el servidor no escolta en el 22 que és el per defecte)
-C : comprimeix les dades abans de transmetre-les (optimitza ample de banda però afegeix feina a la CPU)
-c algo1,algo2,... : indica els algoritmes d'encriptació a usar (per ordre de preferència) durant la comunicació
-X : realitza una redirecció X (veure més avall)
-Y : realitza una redirecció X "trusted" (veure més avall)
-i rutadelarxiudeclaupúblicaquesutilitzarà: (en comptes del ~/.ssh/id_dsa.pub, que és la ruta per defecte)
-v : mode verbós per veure els passos de la comunicació (-q seria el contrari). Amb -vv és més verbós
-V: mostra versió del client
-o opcio_de_fitxer_de_configuració=valor

L'arxiu de configuració és /etc/ssh/ssh_config (general per tots els usuaris), o bé /home/user1/.ssh/config (particular per un usuari en concret; sobreescriu el general); concretament, la preferència dels paràmetres sempre és: línia d'ordres -> fitxer config personal -> fitxer config global. L'ordre en què apareixen escrites les línies dins del fitxer de configuració és important: si hi ha vàries directives iguals només es llegeixen els valors de la primera que apareix. Les directives més importants són:

Port nºport : diu a quin port es conectarà per defecte, si el servidor no escoltés al 22.

Compression yes/no: comprimeix les dades abans de transmetre-les. El nivell de compressió ve donat per la directiva CompressionLevel (1 fast, 9 best). Això funcionarà sempre i quan el servidor accepti aquesta configuració

Ciphers algo1,algo2: indica els algoritmes d'encriptació a usar (per ordre de preferència) durant la comunicació

ForwardX11 yes/no : si val “yes”, no caldrà indicar explícitament el paràmetre -X perquè s'usarà per defecte

PasswordAuthentication yes/no : ofereix la possibilitat d'utilitzar aquest mètode d'autenticació
PubkeyAuthentication yes/no : ofereix la possibilitat d'utilitzar aquest mètode d'autenticació
PreferredAuthentications publickey,password,...: el 1r valor és el mètode preferit; si no va, es prova el següent
IdentityFile... : especifica la ruta de la clau privada de l'usuari (~/.ssh/id_rsa ó id_dsa)

User usuariausarperdefecte (per no haver-lo d'indicar a la línia de comandes)
CheckHostIP yes/no : xequejarà l'autenticitat del servidor consultant a l'arxiu known_hosts
StrictHostKeyChecking yes/no/ask:si és yes, no s'afegeix mai res a known_hosts (s'ha de fer amà)

Host * (las directivas que hay por debajo afectarán solo cuando se conecte al host especificado. Pueden ponerse varias líneas Host cada una diciendo una ip concreta del servidor al que afectarán las directivas posteriores, hasta llegar al final del fichero o bien a otra directiva Host. Debido al orden de lectura de las directivas, se recomienda escribir primero hosts específicos y luego la general con el *. Se puede usar también el ?

ServerAliveInterval nºsegons : envia cada cert temps un paquet i evita així que es tanqui la conexió per inactiva. A més està ServerAliveCountMax

ConnectionAttempts nºintents (abans de deixar-ho estar). Cada intents es fa un cop per segon.

SendEnv var1 var2: envia al shell que s'obrirà al servidor el valor de les variables d'entorn del shell client especificades. El servidor les haurà de reconèixer i acceptar mitjançant la seva directiva AcceptEnv var1 var2

BatchMode yes/no : si val “no”, no es demana contrasenya/passphrase. Aquest mode és útil per ficar-ho en

shell scripts automàtics que no obliguin a què hi hagi una persona al davant de la màquina. Seria gairebé similar a PasswordAuthentication=no

LocalCommand /ruta/comanda/local/que/sexecuta/quan/es/conecta/al/servidor (es pot usar %d -carpeta home local-, %h -host remot-, %l -host local- , %r -remote user name- , %u -local user name- Només funciona si PermitLocalCommand val yes.

Per veure més directives o més informació sobre les anteriors, consultar *man ssh_config*

Client SSHFS (muntatge d'unitats remotes):

Al client només cal executar (prèvia instal.lació del paquet "sshfs"):

sshfs user2@ipServ:/ruta/carpeta/remota /ruta/carpeta/local -p nºport -o uid=nºuidlocal, gid=nºgidlocal

on les opcions uid i gid indiquen que el propietari dels fitxers accessibles a /ruta/carpeta/local passin a ser l'especificat pels valors uidlocal/gidlocal , el qual representaran un usuari local en comptes de user2. Això és per evitar inconsistències en els permisos (si no es fes així, estaríem veient en un punt de muntatge local uns fitxers propietat d'un altre usuari no existent al sistema).

Al client, para desmuntar cal fer: *fusermount -u /ruta/carpeta/local*

Per a què el punt de muntatge sigui permanent cal afegir la següent línia al fitxer /etc/fstab:

sshfs#user2@ipServ:/ruta/remota /ruta/local fuse auto,user,uid=º,gid=nº 0 0

NOTA: En las opciones no se pone “defaults”, porque equivale a: rw, auto, **nouser**, ... con lo que no dejaríamos que un usuario normal montara la carpeta. Esto no sería necesario si fstab funcionara bien, pero resulta que si no tenemos configuradas la autenticación por clave pública, necesitaremos introducir una contraseña en el arranque del sistema que no nos pregunta, así que el montaje no se hace, y deberá culminarlo el usuario local ejecutando "mount /ruta/local"

Client SCP (copia remota):

El client scp ja ve integrat dins del paquet ssh client. Bàsicament permet dues accions:

*"Pujar" arxius locals a una carpeta remota: *scp /ruta/arxius user2@ipServ:/ruta/carpeta*
NOTA: Si no s'indica carpeta remota, per defecte és la HOME de user2

*"Baixar" arxius remots a una carpeta local: *scp user2@ipServ:/ruta/arxius /ruta/carpeta*

Paràmetres interessants que té:

-P nºport
-r : copia recursiva (si com a primer paràmetre es posa la ruta d'una carpeta en comptes de la d'un/s arxiu/s)
-p : manté el propietari, permisos i dates de {c|m|a} originals
-l nºkb/smáximsamplebandapermesos

Client SFTP (còpia remota):

El client sftp ja ve integrat dins del paquet ssh client. No obstant, per a què funcioni el servidor ha d'estar configurat adientment (veure directiva Subsystem). Per executar-lo cal escriure *sftp user2@ipServ:/ruta/carpeta* Si no s'escriu la carpeta on es vol entrar, per defecte és la HOME de user2.

Les comandes de sftp són les mateixes que les d'un client Ftp normal:

cd (en local: *lcd*)	*pwd* (en local: *lpwd*)	*ls* (en local: *lls*)	*mkdir* (en local *lmkdir*)	*rmdir*
rm	*get*	*put*	*progress*	*exit/quit*
!comandaShell				

En aquest cas, els paràmetres s'indicaran com a valors del opció -o, així: *sftp -o Port=1234 user2@192.168.1.1* , per exemple.

A més del client de consola també es pot fer servir el Nautilus (el gestor de fitxers de Gnome) com a client Sftp si s'escriu a la seva barra de navegació el següent: *sftp://user2@ipServ* (o bé *ssh://user2@ipServ).* També es pot fer servir l'opció "Altres ubicacions"->"Conectar al servidor..."

Claus de màquina:

A la instal.lació dels paquets es generen el parell de claus pública/privada dins de /etc/ssh que identifiquen la màquina, i que a cada comunicació serviran per a generar al seu torn una clau de sessió única entre les dues màquines, de manera que no hi pugui haver cap màquina impostora. Normalment el parell pub/priv es crea mitjançant l'algoritme RSA i, per tant, la seva ruta sol ser, respectivament, /etc/ssh/ssh_host_rsa_key.pub i /etc/ssh/ssh_host_rsa_key.

La idea és que la clau pública del servidor -que depén entre altres coses de la seva IP-, es gravarà (després de preguntar primer) en els clients a l´arxiu /home/user1/.ssh/known_hosts la primera vegada que aquests es conectin, per assegurar-se en posteriors connexions que el servidor no sigui un impostor. És a dir: servidor ---clau pública---> client, encara que això també passa de forma recíproca. Si algú és molt paranoic i no es fia d´aquesta primera vegada, sempre pot copiar la clau pública del servidor en un pendrive i copiar-la "manualment" al client., i així ja es tindran i no es preguntarà res la primera vegada.

Es pot fer servir *ssh-keygen -R [ip|host]* que esborra de l'arxiu .ssh/known_hosts la clau corresponent a la IP/host indicat

Claus d'usuari:

Existeix una altra forma de loguejar-se en el servidor Ssh que implica no haver d´escriure user/password cada cop. Es tracta d´utilitzar un parell de claus pública/privada d´usuari. La idea és que cada usuari generi el seu parell propi, i col.loqui la seva clau pública al servidor (la privada romandrà a la màquina client, protegida per contrasenya). Sent user1 al client, volem connectar-nos com user2 a la màquina servidora. Per fer això, a la màquina client s´ha de crear un parell de claus per user1, així:

ssh-keygen -b 1024 -t rsa [-f nomarxiuclaus]

on -b és el nº de bits de la clau (pot ser 512, 1024 o 2048) , -t és l'algoritme criptogràfic que es farà servir per crear-la i -f serveix per indicar un altre nom als fitxers generats diferents dels per defecte. Aquesta comanda pregunta interactivament una "passphrase", que representa una contrasenya incrustada dins de la clau privada per protegir el seu ús si algú la roba; no obstant, si es posa, ens la preguntarà cada cop que ens volguem conectar.

> NOTA: Es pot canviar un passphrase un cop estigui definit, amb aquesta comanda: *ssh-keygen -p -q -t rsa -f ruta/id_rsa.pub -P passantic -N passnou* El paràmetre -q evita els missatges per la sortida estàndar. El paràmetre -p es per canviar la passphrase sense crear un nou fitxer de clau privada.

La comanda anterior crea dos arxius en /home/user1/.ssh que són id_rsa i id_rsa.pub. El següent pas és ara copiar a la màquina servidora, dins de l´arxiu /home/user2/.ssh/authorized_keys, el contingut de l´arxiu id_rsa.pub. Per fer això, es pot fer de vàries maneres, a escollir:

cat ~/.ssh/id_dsa.pub | ssh user2@ipServ "cat - >> ~/.ssh/authorized_keys"
scp ~/.ssh/id_dsa.pub user2@ipServ:.ssh/authorized_keys (si només farem servir aquest client)
ssh-copy-id -i ~/.ssh/id_dsa.pub user2@ipServ

Els permisos de la carpeta ~/.ssh del servidor han de ser 700 i els de l´arxiu authorized_keys han de ser 600.

A més a més, al servidor s´ha de configurar una sèrie de coses per a què això funcioni. En concret, al seu arxiu de configuració s´han d´escriure els següents valors:

PermitRootLogin without-password (o "no", directament)
PasswordAuthentication no (no s'utilitzarà ja més el sistema d'autentificació per contrasenyes)
PubkeyAuthentication yes (activem el sistema d'autentificació per claus)
RSAAuthentication yes (activem explícitament la capacitat de gestionar claus de tipus RSA)
AuthorizedKeysFile %h/.ssh/authorized_keys (indica l'arxiu on es guarden les claus públiques dels clients)

Anell de claus:

Per poder fer servir passphrases i poder fer que el procés d'autenticació continuï sent totalment automàtic, podem fer servir el dimoni ssh-agent. Ssh-agent normalment és un programa que a les distribucions actuals es posa en marxa automàticament amb l'entorn gràfic (X o Wayland), fent així que tots els programes gràfics tinguin una conexió amb ssh-agent en segon pla. Però el podríem executar nosaltres directament, i fins i tot li podríem associar un programa on hi estigués vinculat (i tot els programes fills d'aquest). Per exemple: *ssh-agent gnome-terminal &*

Que Ssh-agent estigui funcionant vinculat a algun entorn (X, gnome-terminal, etc) implica que dins d'aquest entorn es podrà usar el programa *ssh-add* per afegir la passphrase una vegada a l´agent i l´agent passarà al seu torn aquesta informació d´autenticació automàticament cada cop que es necessiti aquest passphrase. Així que el pròxim cop que s´executi el client ssh no preguntarà ni tan sols el passphrase. Suposant que ssh-agent funciona (ps auxw | grep "ssh-agent") , la manera d´afegir la clau a l´anell és executant *ssh-add* , a seques. Si el programa troba la clau RSA, preguntarà la passphrase, i ja està.

Ssh-add té paràmetres interessants:

nomclaupúblicaconcretaaafegir
-t nºsegons : temps que romandrà la clau dins de l'anell (per defecte, per sempre)
-l : llista les claus que estan dins l´anell
-L : per veure si ssh-agent està funcionant o no
-d nomclau : esborra la clau indicada de l´anell
-D : esborra totes les claus de l´anell

Redirecció X11:

Si quisiéramos conectarnos a un servidor Ssh para lanzar allí clientes X sobre nuestro servidor X local, se ha de utilizar el parámetro -X (ó -Y) del cliente Ssh. No obstante, esto solo funciona si el servidor se configura convenientemente mediante las siguientes directivas de su fichero sshd_config:

X11Forwarding yes
X11DisplayOffset 10
X11UseLocalhost yes
AllowTCPForwarding yes #posibilita la creación de túneles Ssh.

NOTA: Si no aparecen alguna de las líneas anteriores, en principio no hará falta ponerlas: sólo cuando estén y tengan otro valor habrá que cambiarlas.

NOTA: Si no se quiere tener que ejecutar todo el rato el parámetro -X en el cliente, en su archivo de configuración se pueden poner las opciones ForwardX11 yes y ForwardAgent yes (esta última sirve para que todos puedan usar el ssh-agent), y será como si siempre se pusiera ese parámetro por defecto.

Tunelling:

Continuarà...