HowTo ASIX Tunnel SSH

Curs 2019-2020

Objectius

1. Túnel SSH directe
   a. El model de funcionament dels Túnels SSH
   b. Túnel SSH directe host to destí (simple)
   c. Túnel SSH directe host to destí to remote

2. Túnel SSH Reverse
   a. El mpdel de funcionament dels túnels Reverse.
   b. Túnel reverse host to destí (simple).
   c. Túnel reverse host to destí to remote.

3. Exemples implementats amb:
   a. Accés web
   b. Accés serveis xinetd: daytime, echo.
   c. Accés a SSH.
   d. Accés PostgreSQL.

Documentació

Aquest document ha estat elaborant utilitzant com a eina de treball un sistema GNU/Linux Fedora 20.

- Documentació de les pàgines man de les ordres.
- Fedora Documentation, Fedora 21. Sistem Administration Guide: 7. OpenSSH

- Observar els gràfics descriptius de les situacions de: https://chamibuddhika.wordpress.com/2012/03/21/ssh-tunnelling-explained/

- http://ubuntuguide.org/wiki/Using_SSH_to_Port_Forward
- https://help.ubuntu.com/community/SSH/OpenSSH/PortForwarding

Ordres:

ssh	ssh-add	ssh-agent	ssh-copy-id	sshd	sshd-keygen
sshfs	ssh-keygen	ssh-keyscan			

Tunnel SSH

Descripció

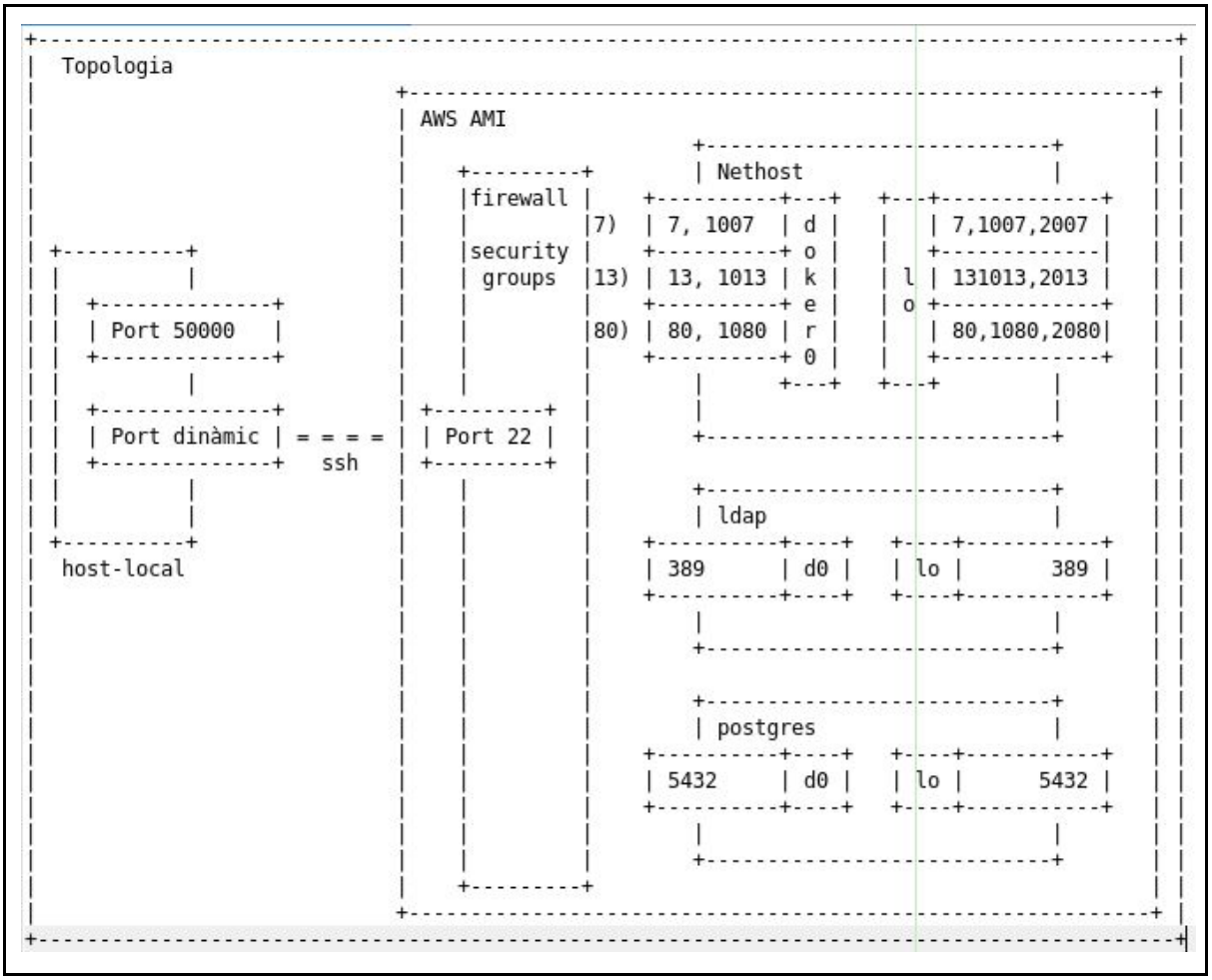
Per entendre bé el funcionament dels túnels SSH cal entendre bé la topologia amb la que volem treballar i el tipus de túnel que volem crear. Aquest apartat intenta fer un resum del què us trobareu al llarg del document.

Què és? Per a què serveix?

Un túnel ssh directe és aprofitar una connexió SSH segura per obrir un port en el host local que connecta amb un port d'un servei en un host-remot, servei al que no es tindria accés directament (generalment). El tràfic dins del SSH és segur. També s'utilitza per accedir a un servei insegur (text plan) però a través d'un canal segur (la connexió ssh).

Parlant clar, obre al host local una porta dimensional estelar tipus stargate o star-treck per connectar a través de l'hiperespai (ssh) amb un destí remot que generalment no seria accessible). Podeu tenir al host de l'aula una porta secreta intergalàctica que us fa apareixer a un servei de un dels ordinadors que teniu a casa.

Exemple de topologia:



Elements de la topologia:

- ***host-local***: host des del que establim el túnel ssh. Host des d'on treballem i realitzem la ordre ssh per connectar creant un túnel al host destí. Per exemple el nostre ordinador de casa fent un ssh al router de l'empresa o al router de l'escola.

- ***host-destí***: anomenem host destí al host al que ens connectem via ssh. Quan des del host client fem la ordre ssh per crear el túnel ens cal, com a tota ordre ssh, posar el *user@hostdesti* al que volem connectar. Lògicament cal disposar d'un compte d'usuari (amb el que volem connectar) en el host destí i aquest ha de tenir el servei sshd engegat.
  Exemples de host destí poden ser el router de l'empresa al que volem connectar, el router de l'escola al que volem connectar o la AMI de AWS EC2 al que volem connectar via ssh.

- ***host-remot***: Usualment el host-destí és un router o punt d'entrada amb el servei sshd engegat que governa una xarxa local privada. Qualsevol host que està dins d'aquesta xarxa privada remota és un host remot.
  Des del host-local és impossible accedir per internet al host-remot.

Un exemple és que des de casa (host-local) volem accedir a través del router de l'empresa (host-destí) a un dels hosts de dins de l'empresa (host-remot).
Un exemple més pràctic per vosaltres és des de l'aula (host-local) accedir per ssh al la AMI de AWS EC2 (host-destí) per ***'rebotar'*** a un servei ofert per un dels containers que s'executen dins de la AMI (hosts-remots), serveis no mapejats a la AMI.

Model de l'aula:

- **host-local**: el vostre host de l'aula
- **host-destí**: la AMI de AWS EC2
- **host-remot**: els hosts que estan en la xarxa intern privada mynet desplegats amb docker dins de la AMI i que NO fan map dels seus ports a la AMI.
- Observeu que amb aquest model des de l'aula no hi ha manera d'accedir ls serveis dels containers, perquè són serveis locals a la xarxa mynet.

Tipus de túnels

Una altra cosa que espero que us ajudi a entendre el model dels túnels és descriure anticipadament els tipus de túnels que hi ha.

- ***Túnel directe a host-destí***: Aquest és el tipus més usat i consisteix en accedir des del host-local a un servei del host-destí que no està publicat de cara a l'exterior. És un servei que només s'hi pot accedir des del host-destí directament.
  En aquest model des del host-remot obrim una porta local (al host-local) que permet accedir al port del servei del host-destí. Un port al que no hi podem accedir directament perquè no és públic.
  El túnel s'estableix entre host-local i host-destí i en el destí *rebota* al *localhost* al port del servei al que es vol accedir.
  Els exemples més típics són per accedir a serveis del host-destí que no permeten connexions segures (un postgres, hhtp-80, ldap-389, etc) i que no es pubiquen a l'exterior per evitar-ne l'accés. Només són accessibles des del propi host-destí.
  Observeu que en aquest model el 'rebot' és al propi host-destí, indicat com a localhost.

- ***Túnel directe a host-destí per accedir a host-remot-destí***: En aquest model des del host-local volem accedir a un servei que està en un host-remot al qual no es té access. S'aconsegueix generant una porta local al host-local que connecta directament al port del servei del host-remot. però com ho fa si no hi té accés?
  Cal fer un túnel ssh del host-local al host-destí (sempre són així els túnels) i indicar el *'rebot'* al *host-remo*t.
  Per exemple des de l'aula (host-local) fem un túnel ssh a la AMI de AWS EC2 (host-destí) i indiquem el *rebot* al container Docker anomenat *ldap.edt.org al port 389*. Aquest container està en la xarxa interna local de docker dins de AWS EC2 i no s'hi podria accedir des de l'exterior de la AMI si no fos per el túnel.

- ***Túnel Reverse al host-destí***: Consisteix en fer exactament el contrari que un túnel directe al host-destí. Es tracta de establir un túnel del host-local al host-destí que deixa una porta oberta en el destí per accedir al servei indicat en el host-origen.
  Easy-peasy!
  Des de casa no podeu accedir al host de l'aula perquè hi ha el firewall de l'escola que impedeix que podeu fer connexions ssh a dins de l'escola. Però… podem fer alguna ***trampa*** per deixar una porta oberta per accedir des de l'exterior al host de l'aula?. (nota: recordeu que imcomplir les normes d'utilització dels equips informàtics de l'escola està penat amb sedicií, rebelió i esquarteració).
  Imaginem que voleu deixar a la màquina AMI una porta oberta per accedir al host de l'aula. Així des de la AMI podreu fer ssh a un port local de la AMI i apareixereu per art de màgia (màgia=túnel ssh reverse) al host de l'aula.
  El procediment és des de l'aula (host-local) connectar establint un túnel ssh revers a la AMI de AWS EC2 (host-destí) i deixant allà obert un port arbitrari (per exemple el 9000) que connecta amb el servei del host-local que volem fer accessible des de l'exterior (per exemple ssh).

  El túnel reverse sempre obre una porta en el host-destí. En aquest exemple l'obre per accedir a un servei del host-origen.

  Un altre exemple (més rebuscat) és permetre a tothom veure amb el phpldapadmin que està a AWS EC2 (amb mapejat de ports). Aquest phpldapadmin obté les dades del ldap que tenim executant a l'aula, com? Doncs per al phpldapadmin és com si el servidor ldap estigués a la propia AMI en un port local. Per fer-ho des del host de l'aula (host-local) s'ha obert un port amb un túnel ssh reverse dins de la AMI AES EC2 que connecta amb el servei ldap del host de l'aula (atenció, cal que el server ldap publiqui el seu port 389 al host de l'aula).

- ***Túnel Reverse al host-remot-origen***: Altre cop el contrari del túnel directe a un host remot. Atenció que aquí el model mental 'casca' unes quantes neurones!.
  En aquesta topologia tenim una xarxa local (origen) amb diversos hosts un d'ells és el host-local des d'on s'establirà el túnel. És vol obrir una porta a un host extern, el host-destí, per accedir a un servei que està en la xarxa origen, però no en el host-local. Observeu que el 'rebot' és en la xarxa origen i no en la xarxa destí.
  Per exemple des de l'aula volem deixar una porta oberta a la AMI (host-destí) des la qual es podrà accedir a un altre host de l'aula (host-remot-origen).
  Un altre exemple és que en el nostre host de l'aula (host-local) tenim vàris containers Docker executant-se en una xarxa privada interna que no són accessibles des de l'exterior (aquests containers docker són els hosts-remot-origen). Podem establir des del host-local de l'aula un túnel reverse a la AMI (host-destí) deixant allà obert un port per el qual es podrà accedir a un servei dels dockers, per exemple al servei 389 del docker ldap.edt.org (host-remot-origen).
  Facile e divertente!

- ***Ports túnel SSH***: atenció, la configuració per defecte del servidor SSH crea els ports dels túnels amb un bind únicament al localhost. Per tant, aquesta porta que obren els túnels ssh és només accessible al localhost del host on s'ha creat.
  Si es vol que el port (la porta que s'ha obert) es publiqui per altres interfícies cal modificar la contiguració del SSH.
  També cal modificar el format de les ordres dels túnels ssh incloent el bind a fer en l'establiment del port.

Format i significat de l'ordre

-L [bind_address:]portorigen:hostremot:hostremotport	user@hostdestí
-R [bind_address:]portdesti:hostremotorigen:hostremotport	user@hostdestí

Túnel directe:

- El format usual que trobem més és: *5000:localhost:13 user@hostdestí*. Significa crear un túnel ssh connectant com a user al hostdestí. Obrir el port 5000 (en el host-local origen) que connecta directament al port 13 del host remot (localhost). Com que el host remot és localhost significa que el port 13 és al mateix host-destí on hem connectat.
  Obro dins de casa a la meva habitació una porta que em condueix directament al host-destí.
- El format ssh amb 'rebot' és: *5000:ldap.edt.org:389 user@hostdestí*. Significa crear un túnel ssh connectant com a user al hostdestí. Obrir el port 5000 en el host-local (l'origen des d'on fem la ordre) que connecta directament al port 389 del host-remot ldap.edt.org.
  Observeu que la connexió ssh la fem de host-local a host-destí (on hi ha el servei sshd engegat i cal accedir a un compte d'usuari vàlid), però el redireccionament del port 5000 no es queda al host-destí sinó que rebota i accedeix al port 389 d'un altre host (host-remot).
  Això permet accedir a un host-remot que segurament les regles dels firewalls no permetrien.
  obro a casa meva a la meva habitació una porta que em condueix directament a un host-remot (només accessible via el host-destí).
- El format més complexe és: *0.0.0.0:5000:ldap.edt.org:389 user@hostdestí*. Aquí la diferència és el binding del port 5000 que es fa en el host-local. Per defecte els ports SSH s'obren només en el localhost, però es pot indicar opcionalment com a primer argument a quina de les interfícies volem fer bind. Si s'utilitza 0.0.0.0 significa totes les interfícies del host-local.

Túnels reverse

- El format usual que trobem més és: *5000:localhost:13 user@hostdestí*. Significa crear un túnel ssh connectant com a user al hostdestí. Obrir el port 5000 en el host-destí que permet accedir directament al port 13 del host-local (l'origen de l'ordre

ssh). Com que el host-remot-origen és localhost significa que el port 13 és el del mateix host-local des d'on s'ha realitzat l'ordre.
Obro a fora una porta per entrar a casa meva a la meva habitació.

- El format ssh amb 'rebot' és: *5000:ldap.edt.org:389 user@hostdestí*. Significa crear un túnel ssh connectant com a user al hostdestí. Obrir el port 5000 en el host-destí que connecta directament al port 389 del host-remot-origen ldap.edt.org.
  Observeu que la connexió ssh la fem de host-local a host-destí (on hi ha el servei sshd engegat i cal accedir a un compte d'usuari vàlid), però el redireccionament del port 5000 no es queda al host-local sinó que rebota i accedeix al port 389 d'un altre host (host-remot-origen). Aquest és un host que hi ha a la xarxa del host-local.
  Obro a fora una porta per entrar a casa meva però a una altra habitació de les de casa meva (no a la meva).
- El format més complexe és: *0.0.0.0:5000:ldap.edt.org:389 user@hostdestí*. Aquí la diferència és el binding del port 5000 que es fa en el host-destí. Per defecte els ports SSH s'obren només en el localhost, però es pot indicar opcionalment com a primer argument a quina de les interfícies volem fer bind. Si s'utilitza 0.0.0.0 significa totes les interfícies del host-destí.

Tunnel SSH

man ssh

TCP FORWARDING
Forwarding of arbitrary TCP connections over the secure channel can be specified either on the command line or in a configuration file. One possible application of TCP forwarding is a secure connection to a mail server; another is going through firewalls.

In the example below, we look at encrypting communication between an IRC client and server, even though the IRC server does not directly support encrypted communications. This works as follows: the user connects to the remote host using ssh, specifying a port to be used to forward connections to the remote server. After that it is possible to start the service which is to be encrypted on the client machine, connecting to the same local port, and ssh will encrypt and forward the connection.

The following example tunnels an IRC session from client machine "127.0.0.1" (localhost) to remote server "server.example.com":

```
$ ssh -f -L 1234:localhost:6667 server.example.com sleep 10
$ irc -c '#users' -p 1234 pinky 127.0.0.1
```

This tunnels a connection to IRC server "server.example.com", joining channel "#users", nickname "pinky", using port 1234. It doesn't matter which port is used, as long as it's greater than 1023 (remember, only root can open sockets on privileged ports) and doesn't conflict with any ports already in use. The connection is forwarded to port 6667 on the zremote server, since that's the standard port for IRC services.

The -f option backgrounds ssh and the remote command "sleep 10" is specified to allow an amount of time (10 seconds, in the example) to start the service which is to be

tunnelled. If no connections are made within the time specified, ssh will exit.

-L [bind_address:]localport:hostremote:hostremoteport user@sshserver

Specifies that the given port on the local (client) host is to be forwarded to the given host and port on the remote side. This works by allocating a socket to listen to port on the local side, optionally bound to the specified bind_address. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to host port hostport from the remote machine. Port forwardings can also be specified in the configuration file. IPv6 addresses can be specified by enclosing the address in square brackets. Only the superuser can forward privileged ports.

By default, the local port is bound in accordance with the GatewayPorts setting. However, an explicit bind_address may be used to bind the connection to a specific address. The bind_address of “localhost” indicates that the listening port be bound for local use only, while an empty address or ‘*’ indicates that the port should be available from all interfaces.

X11 FORWARDING
If the ForwardX11 variable is set to “yes” (or see the description of the -X, -x, and -Y options above) and the user is using X11 (the DISPLAY environment variable is set), the connection to the X11 display is automatically forwarded to the remote side in such a way that any X11 programs started from the shell (or command) will go through the encrypted channel, and the connection to the real X server will be made from the local machine. The user should not manually set DISPLAY. Forwarding of X11 connections can be configured on the command line or in configuration files.

The DISPLAY value set by ssh will point to the server machine, but with a display number greater than zero. This is normal, and happens because ssh creates a “proxy” X server on the server machine for forwarding the connections over the encrypted channel.

ssh will also automatically set up Xauthority data on the server machine. For this purpose, it will generate a random authorization cookie, store it in Xauthority on the server, and verify that any forwarded connections carry this cookie and replace it by the real cookie when the connection is opened. The real authentication cookie is never sent to the server machine (and no cookies are sent in the plain).

If the ForwardAgent variable is set to “yes” (or see the description of the -A and -a options above) and the user is using an authentication agent, the connection to the agent is automatically forwarded to the remote side.

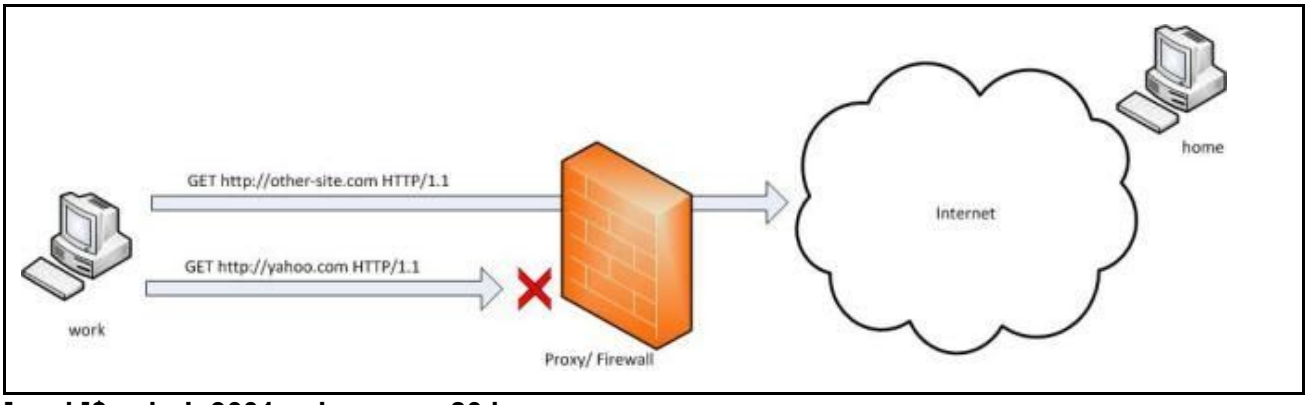
Exemples documentació externa

Exemples extrets del document “SSH Tunneling Explained” de la web “https://chamibuddhika.wordpress.com ”.

Per poder realitzar túnels SSH, tant directes com reverse, cal des d'un host poder realitzar una connexió ssh a un altre host, on evidentment hi ha engegat un servidor ssh.

Exemple 1:

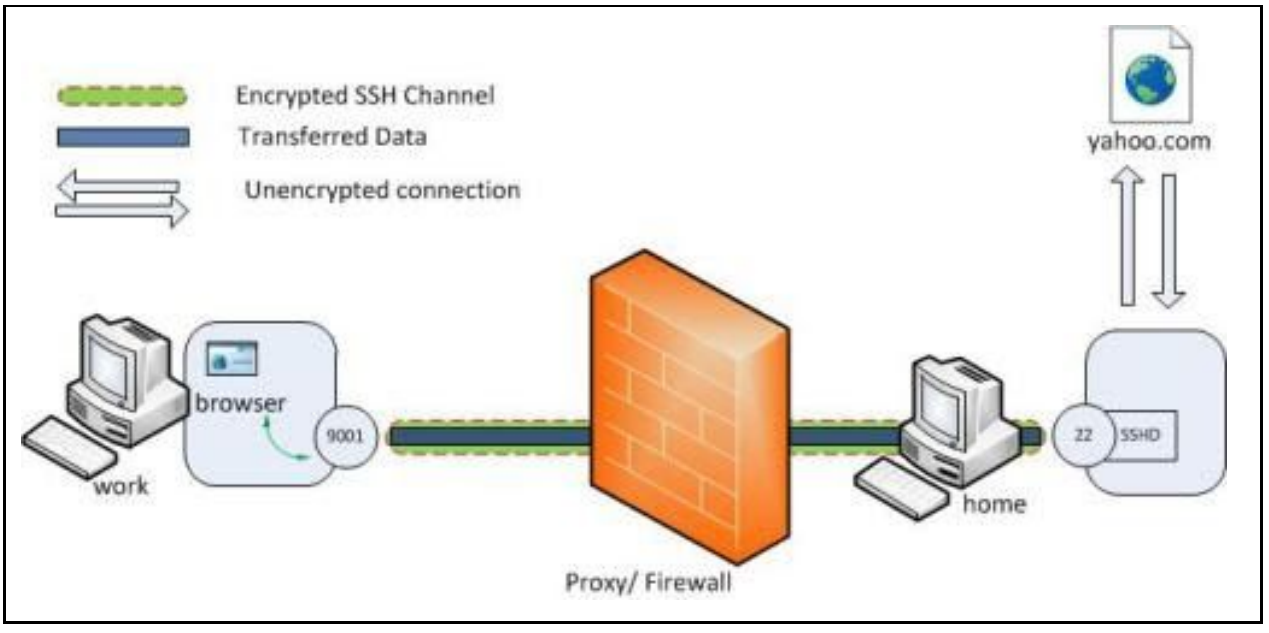
Crear un túnel del host work al host home, fent que aquest es connecti al port 80 de yahoo.com.



[work]$ ssh -L 9001:yahoo.com:80 home

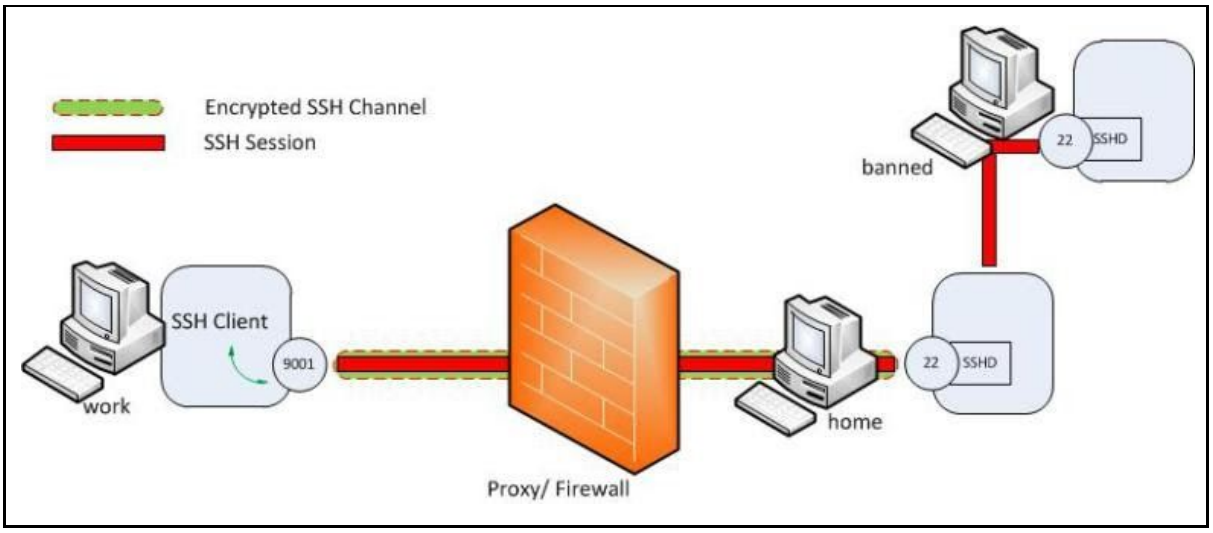
D'aquesta manera accedint al port local 9001 del host work en realitat s'accedeix al port 80 de yahoo.com, a través de home:

[work]$ telnet localhost 9001
GET / HTTP/1.0



Exemple 2:

En aquest exemple es realitza una connexió del ssh de work al ssh del host banned, a través del host home, ja que el firewall no permet l'accés directe de work a banned.



[work]$ ssh -L 9001:banned:22 work@home

Ara des del host work si es realitza una sessió ssh al propi port 9001 en realitat s'està accedint al ssh del host banned.

[work]$ ssh -p 9001 localhost

Exemples @edt

Exemple 3:

client ------> desti (ssh-server, web-server)

Establir una connexió al port 80 del host destí, fent passar el tràfic insegur HTTP per el túnel SSH . En el host client s'obrirà un port local 3030 que portarà el tràfic HHTP per dins del túnel fins al port 22 del host destí. En el destí el dimoni SSH encaminarà aquest tràfic al port 80 del propi destí.

En el destí CAL que s'executi el servei SSHD (el dimoni) i cal que l'usuari de client pugui establir una sessió SSH al destí.

[client]$ SSH -L 3030:localhost:80 user@desti

Ara en el host client accedint al port 80 local en realitat s'està accedint via el túnel SSH al port 80 del host destí.

[client]$ telnet localhost 3030

__atenció__ localhost no significa el host client sinó el host destí!

Exemple 4:

Ídem anterior fent un túnel SSH per accedir al servei daytime (del xinetd) del host destí. En aquest exemple s'utilitza -f per deixar el túnel SSH en background, executant la comanda sleep. Així no consumeix la consola.

```
[client]$ SSH   -f   -L 3031:localhost:13  user@desti   sleep 100000
[client]$ telnet localhost 3031
```

Exemple 5:

client -- -- -- -- (firewalll) -- -- --- --- --- --> (80) destí
(3032) web server
|----------> (22)SSH-Server ----------(80)
No es pot realitzar una connexió al port 80 del destí (un web server) perquè el firewall ho impedeix. Si es pot accedir al servidor SSH del SSH-Server. I aquest host (el SSH-Server) si que pot creuar el firewall i accedir al host destí al servei web (o al que sigui!).

El túnel SSH consisteix en establir un port local 3032 al host client que connecta via ssh al host SSH-Server (creuant la zona pública o insegura). El servidor SSH genera una connexió al port 80 del destí, és tràfic HHTP insegur, però dins del firewall (que suposem una zona segura).

Per exemple el client és un host en un locutori internet, la connexió SSH és fins al router/firewalll de casa o de l'empresa, que té un servei SSH actiu i on el client es pot connectar. Des d'quest servidor SSH s'estableix una connexió normal a un host de dins de casa o de l'empresa (on el tràfic suposem que és segur!).

```
[client]$ SSH -L 3032:destí:80 user@ssh-server
[client]$ telnet localhost 3032
        GET / HTTP/1.0
```

Observar que 3032 és el port local del client. desti:80 indica el host final al que contactar, a quin host i port el servei SSH reenviarà el que rep de client. user@ssh-server indica a quin host s'estableix el túnel, el tràfic xifrat. Usualment és un host que fa de frontera entre una zona insegura i una zona segura (que pot creuar el firewall) o que fa de router entra una xarxa pública i una de privada.

Exemple 6:

Ídem exemple anterior connectant a un host de classe al servei daytime. Client host i25, servidor SSH host i26 i servei al que accedir (en aquest cas daytime) al host i27:

[i25]$ SSH -f -L 3033:i27:13 user@i26 sleep 100000
[i25]$ telnet localhost 3033

Exemple 7:

Ídem demostrant que és el servidor SSH el que fa la resolució de noms del host destí al que accedir.

- Imaginem un host client en un locutori d'internet (host i26),
- Connectem al router/SSH-Server de casa que té el port SSH obert per establir el túnel (host i26), és una conexió de la xarxa internet, IPs públiques,
- Accedir a un host de dins de casa de la xarxa privada local de casa, a un host anomenat serverJocs (host i27/serverjocs). Aquest nom de host només és vàlid dins de la xarxa local.

Primerament afegir al servidor ssh una entrada al /etc/hosts anomenada serverJocs que identifiqui el host destí (host i27)
[i26]# echo “”192.168.2.57 serverjocs”” >> /etc/hosts

Establir el túnel des del host client (host i25) a un destí que és un nom de host de la xarxa privada local interna de casa (i que evidentment no es resoluble des del locutori on ens estem connectant)

[i25]$ SSH -L 3034:serverjocs:13 user@i26
[i25]$ telnet localhost 3034

Reverse Tunnel

Remote Port Forwarding

Remote port forwarding lets you connect from the *remote* SSH server to another server. To use remote port forwarding, you need to know your destination server, and two port numbers. You should already know your destination server, and for basic uses of port forwarding, you can usually use the port numbers in Wikipedia's list of TCP and UDP port numbers.

For example, say you wanted to let a friend access your remote desktop, using the command-line SSH client. You would use port number 5900 (the first *VNC* port), and destination server *localhost*:

```
ssh -R 5900:localhost:5900 guest@joes-pc
```

The -R option specifies *remote* port forwarding. For the duration of the SSH session, Joe would be able to access your desktop by connecting a VNC client to port 5900 on his computer (if you had set up a shared desktop).

-R [bind_address:]localport:hostremote:hostremoteport user@sshserver

Specifies that the given port on the remote (server) host is to be forwarded to the given host and port on the local side. This works by allocating a socket to listen to port on the remote side, and whenever a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to host port hostport from the local machine.

Port forwardings can also be specified in the configuration file. Privileged ports can be forwarded only when logging in as root on the remote machine. IPv6 addresses can be specified by enclosing the address in square braces.

By default, the listening socket on the server will be bound to the loopback interface only. This may be overridden by specifying a bind_address. An empty bind_address, or the address '*', indicates that the remote socket should listen on all interfaces. Specifying a remote bind_address will only succeed if the server's GatewayPorts option is enabled (see sshd_config(5)).

If the port argument is '0', the listen port will be dynamically allocated on the server and reported to the client at run time. When used together with -O forward the allocated port will be printed to the standard output.

Exemples Reverse Tunnel

Exemples extrets del document "SSH Tunneling Explained" de la web "https://chamibuddhika.wordpress.com ".

Per poder realitzar túnels SSH, tant directes com reverse, cal des d'un host poder realitzar una connexió ssh a un altre host, on evidentment hi ha engegat un servidor ssh.

Exemple 1:

Des de dins d'una organització o xarxa local s'estableix un túnel ssh cap a un altre host per obrir-hi en aquest client extern una 'porta' per on accedir a aquesta xarxa interna/local. Usualment els firewalls filtren el tràfic d'entrada però permeten tràfic de sortida. Per exemple usualment està filtrat el tràfic SSH entrant però es permet el tràfic SSH sortint.

Establir un túnel invers des del host de treball de casa (host i27) a un host extern públic (host i26). El host de casa (host i27) té engegat el servei daytime (port 13). A aquest servei daytime s'hi pot accedir des de dins de la xarxa interna de casa, però no de fora. Amb un túnel reverse ssh es deixarà una porta oberta perquè des de l'exterior es pugui accedir a un servei (el daytime) de la xarxa interna privada de casa. Tot el que cal és permetre el tràfic ssh.

[i27]$ SSH -R 3035:localhost:13 user@i26

Així des del host privat (host i27) s'estableix un túnel cap enfora al host públic (host i26). En aquest host i26 es crea un port 3035 que permet accedir al servei daytime, port 13 del host privat i27. En connectar a aquest port 3035 local del i26 s'obté l'hora proporcionada per el servei daytime del host i27. Sense el túnel reverse el firewall de la xarxa local de casa no permetria l'accés al servei daytime del i27.

[i26]$ telnet localhost 3035

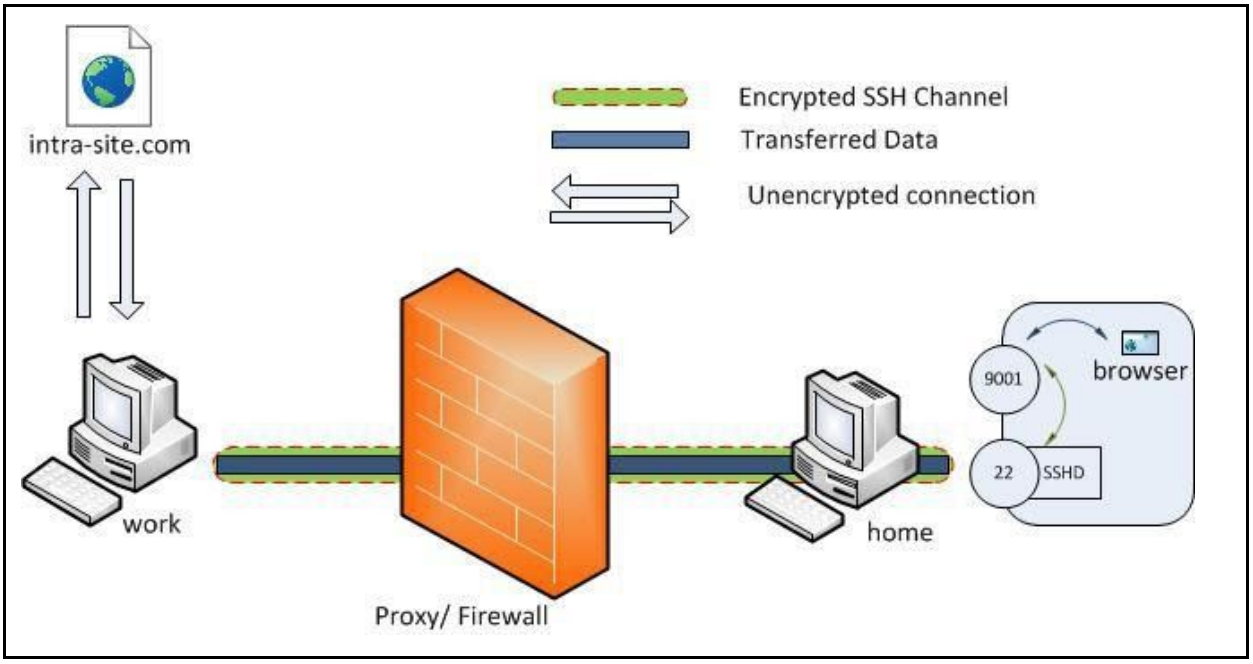
Exemple 2:

En aquest exemple des del host work (dins de la zona protegida del firewall) es crea un túnel invers per permetre al host home accedir al servei de intranet, que no es accessible des de fora perquè el firewall no ho permet.

En el host home es pot accedir localment al port 9001 i en realitat s'està accedint a la intranet de dins del firewalll.

[work]$ ssh -R 9001:intra-site.com:80 user@home

[home]$ telnet localhost 9001



Exemple 3:

Implementar a l'aula un túnel invers per permetre a un client accedir a un servei que està en una zona protegida o en una xarxa privada. és a dir, el client en principi no té accés al servei que proporciona el host destí. Però el host *client SSH* si que té accés als serveis del host destí.

- En una intranet/xarxa-privada hi ha un host (host i27) que proporciona un servei daytime (del xinetd). Aquest servei nomes és accessible per la intranet.
- Un host de la intranet (host i26) fa la funció firewall/router o de intermediari (on està l'usuari) estableix un túnel ssh invers amb el client.
- El host client (host i25) disposarà del port local 3036 que li permetrà l'accés passant pel túnel ssh al servei daytime del host destí, accés que no podria realitzar directament fora del túnel.

Primerament engegar el servei daytime-stream del xinetd en el host destí (host i27) i configurar l'accés a aquest servei únicament per a ell mateix i per al host intermediari (i26), amb la directiva:

only_from = localhost 192.168.2.57 192.168.2.56

Establir des del host intermediari (host i26) el túnel al client (host i25):

[i26]$ SSH -R 3036:i27:13 user@i25

Observem que a l'ordre a part de tenir el -R els ports s'indiquen com usualment, però signifiquen 'el revés'. El port 3036 s'obrirà en el client (user@i25) i connectarà al port 13 del host destí i27.

Dynamic Port Forwarding

Dynamic Port Forwarding

Dynamic port forwarding turns your SSH client into a *SOCKS proxy server*. *SOCKS* is a little-known but widely-implemented protocol for programs to request any Internet connection through a *proxy server*. Each program that uses the proxy server needs to be configured specifically, and reconfigured when you stop using the proxy server.

For example, say you wanted Firefox to connect to every web page through your SSH server. First you would use dynamic port forwarding with the default SOCKS port:
ssh -C -D 1080 laptop

The -D option specifies *dynamic* port forwarding. 1080 is the standard SOCKS port. Although you can use any port number, some programs will only work if you use 1080. -C enables compression, which speeds the tunnel up when proxying mainly text-based information (like web browsing), but can slow it down when proxying binary information (like downloading files).

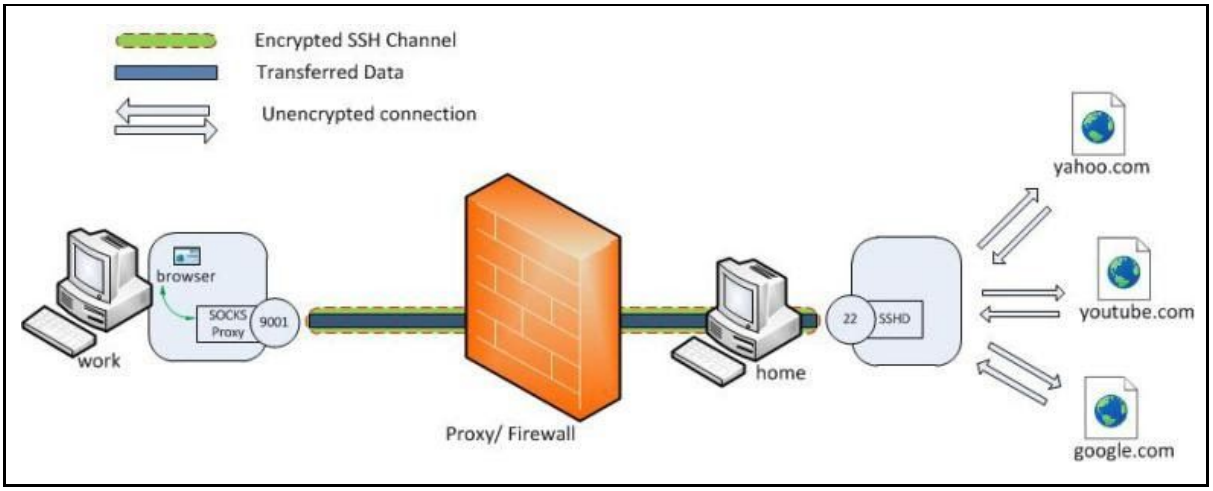
Next you would tell Firefox to use your proxy:

- go to Edit -> Preferences -> Advanced -> Network -> Connection -> Settings...
- check "Manual proxy configuration"
- make sure "Use this proxy server for all protocols" is cleared
- clear "HTTP Proxy", "SSL Proxy", "FTP Proxy", and "Gopher Proxy" fields
- enter "127.0.0.1" for "SOCKS Host"
- enter "1080" (or whatever port you chose) for Port.

You can also set Firefox to use the DNS through that proxy, so even your DNS lookups are secure:

- Type in about:config in the Firefox address bar
- Find the key called "network.proxy.socks_remote_dns" and set it to true

The SOCKS proxy will stop working when you close your SSH session. You will need to change these settings back to normal in order for Firefox to work again.
To make other programs use your SSH proxy server, you will need to configure each program in a similar way.



[work]$ ssh -D 9001 home

SSH VPN

SSH-BASED VIRTUAL PRIVATE NETWORKS

ssh contains support for Virtual Private Network (VPN) tunnelling using the tun(4) network pseudo-device, allowing two networks to be joined securely. The sshd_config(5) configuration option PermitTunnel controls whether the server supports this, and at what level (layer 2 or 3 traffic).

The following example would connect client network 10.0.50.0/24 with remote network 10.0.99.0/24 using a point-to-point connection from 10.1.1.1 to 10.1.1.2, provided that the SSH server running on the gateway to the remote network, at 192.168.1.15, allows it.

On the client:

```
# ssh -f -w 0:1 192.168.1.15 true
# ifconfig tun0 10.1.1.1 10.1.1.2 netmask 255.255.255.252
# route add 10.0.99.0/24 10.1.1.2
```

On the server:

```
# ifconfig tun1 10.1.1.2 10.1.1.1 netmask 255.255.255.252
# route add 10.0.50.0/24 10.1.1.1
```

Client access may be more finely tuned via the /root/.ssh/authorized_keys file (see below) and the PermitRootLogin server option.
The following entry would permit connections on tun(4) device 1 from user “jane” and on tun device 2 from user “john”, if PermitRootLogin is set to “forced-commands-only”:

```
tunnel="1",command="sh /etc/netstart tun1" ssh-rsa ... jane
tunnel="2",command="sh /etc/netstart tun2" ssh-rsa ... john
```

Since an SSH-based setup entails a fair amount of overhead, it may be more suited to temporary setups, such as for wireless VPNs. More permanent VPNs are better provided by tools such as ipsecctl(8) and isakmpd(8).

Exemples 2017-2018

Descripcó:

```
ssh -L portlocal:hostRemot:portRemot user@destí
```

Túnels SSH directe:

Exemple 1:

```
ssh  -L 9001:localhost:13 192.168.1.40 "sleep 123456789"
 telnet localhost 9001
Trying 127.0.0.1...
Connected to localhost.
```

```
        Escape character is '^]'.
        17 MAR 2018 18:48:22 CET
        Connection closed by foreign host.
```

Exemple 2:

```
        (.1.40)  ncat -l 50010
         ssh  -L 9001:localhost:50010 192.168.1.40 "sleep 123456789"
         telnet localhost 9001
        Trying 127.0.0.1...
        Connected to localhost.
        Escape character is '^]'.
        hola que tal
        molt bé
```

Exemple 3:

```
        (1.40)  ncat -l 50010
        (1.40) /etc/hosts → 1.40 hostremot
        ssh  -L 9001:hostremot:50010 192.168.1.40 "sleep 123456789"
         telnet localhost 9001
        Trying 127.0.0.1...
        Connected to localhost.
        Escape character is '^]'.
        hola que tal
        molt be i tu
```

Exemple 4:

```
         ssh  -L 9001:hostremot:22 192.168.1.40 "sleep 123456789"
         telnet localhost 9001
        Trying 127.0.0.1...
        Connected to localhost.
        Escape character is '^]'.
        SSH-2.0-OpenSSH_7.2
```

Exemple 5:

```
         ssh  -L 9001:hostremot:22 192.168.1.40 "sleep 123456789"
        ssh -p 9001 localhost
        The authenticity of host '[localhost]:9001 ([127.0.0.1]:9001)' can't be
        established.
        ECDSA key fingerprint is
        29:36:9a:f9:4e:22:09:04:2a:8c:86:da:32:8b:2a:43.
        Are you sure you want to continue connecting (yes/no)? yes
        Warning: Permanently added '[localhost]:9001' (ECDSA) to the list of
        known hosts.
        ecanet@localhost's password:
        Last login: Sat Mar 17 18:46:39 2018 from 192.168.1.38
        … som a hostremot amb ssh..
```

Exemple 06:

```
        (.1.40) /etc/hosts → ldapserver 172.17.0.2
        (1.40) engegar ldapserver en docker detach
        ssh  -L 9001:ldapserver:389 192.168.1.40 "sleep 123456789"
         ldapsearch -x -LLL -h localhost:9001 -s base -b 'dc=edt,dc=org'
        dn: dc=edt,dc=org
        dc: edt
        description: Escola del treball de Barcelona
        objectClass: dcObject
        objectClass: organization
        o: edt.org
```

Exemple 07:

```
        (1.40) engegar un docker amb postgresql
        (1.40) /etc/hosts → 172.17.0.3 postgres
         ssh  -L 9001:postgres:5432 192.168.1.40 "sleep 123456789"
         psql -h localhost -p 9001 -U postgres
```

```
 psql -h localhost -p 9001 -U postgres -d training -c "select * from
oficinas;"
 oficina |   ciudad      | region | dir | objetivo  |  ventas
---------+-------------+--------+-----+-----------+-----------
        22 | Denver    | Oeste  | 108 | 300000.00 | 186042.00
        11 | New York| Este   | 106 | 575000.00 | 692637.00
        12 | Chicago   | Este   | 104 | 800000.00 | 735042.00
        13 | Atlanta   | Este   | 105 | 350000.00 | 367911.00
        21 | Los Angeles | Oeste  | 108 | 725000.00 | 835915.00
(5 rows)
```

Tunel ssh reverse:

Exemple 1:

Pràctiques 2018-2019

Per fer aquests exercicis necessitem:

- host: un host de l'aula.
- AWS EC2. tres hosts a AWS EC2.
- nethost: imatge docker edtasixm11/net18:nethost.
- postgres: imatge docker
- ldap: imatge docker edtasixm06/ldapserver:18group
- una mica de fairy i ajax!

Recomanació: Crear a AWS EC2 un security group anomenat nethost on obrim tots els ports necessaris:

- echo(7), daytime (13), echo-bis(2007), daytime-bis(2013)
- httpd(80), httpd-bis(2080).
- pop(110), imap(143), pop3s(995), imaps(993)
- ldap (389)q
- postgres(5432)
- Atenció: sshd(1022)

Túnel SSH directe:

Veurem els exemples de:

- ❏ Túnel d'un protocol insegur
- ❏ Servei en el host destí aïllat (bind al localhost)
- ❏ Servei remot a un host-remot de la xarxa destí
- ❏ Visualitzacions externes

Clàssic: túnel d'un protocol insegur

Exemple host a host per accedir a un servei que no és segur utilitzant una connexió ssh segura. Simplement proporciona la seguretat de la connexió segura, una connexió que podríem fer igualment al servidor però amb un protocol insegur.

Exemples amb:

- nethost
- ldapserver
- postgres
- phpldapadmin

Exemple-1 TD-HL-HD. Nethost amb ports mapejats.

Infraestructura: el host de l'aula (host-local) i un container docker nethost (edtasixm11/net18:nethost) en un host AWS EC2 amb els ports mapejats al 7, 13 i 80 (AMI és el host-destí). Obrir el firewall de AWS EC2 al port SSH i als ports indicats. Accedir als serveis echo, daytime i web (un cada cop) del host AWS EC2 usant una connexió segura ssh des del host de l'aula.

Engegar el nethost amb els ports mapejats en el host AWS EC2

```
[fedora@ip-172-31-20-75 ~]$ docker run --rm --name net1     -h net1  --net mynet -p 13:13 -p 7:7 -p 80:80   -d
edtasixm11/net18:nethost
8a90d498ee7ab3dccd734fe7c726d78c0a331112824f7a7472733e85385dfb4c

[fedora@ip-172-31-20-75 ~]$ docker ps
CONTAINER ID      IMAGE                 COMMAND                   CREATED           STATUS              PORTS
            NAMES
8a90d498ee7a      edtasixm11/net18:nethost   "/opt/docker/startup…"   6 seconds ago      Up 5 seconds
0.0.0.0:7->7/tcp, 0.0.0.0:13->13/tcp, 0.0.0.0:80->80/tcp   net1
```

Establir el túnel interactiu al host de AWS EC2.

```
$ ssh -i .ssh/MyKeyPair.pem  -L 50000:localhost:13 fedora@35.178.81.76
Last login: Thu Mar  7 16:37:17 2019 from 2.137.66.200
[fedora@ip-172-31-20-75 ~]$
```

Test des del host de l'aula:

```
$ telnet localhost 50000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
07 MAR 2019 17:11:49 UTC
Connection closed by foreign host.
```

Exemple-2 TD-HL-HD. Consulta ldap insegura usant un túnel ssh segur.

Infraestructura: el host de l'aula (host-local) i un container docker ldapserver (edtasixm06/ldapserver:18group) en un host AWS EC2 amb el port 389 mapejat (AMI és el host-destí). Obrir el firewall de AWS EC2 el port SSH i el port ldap. Accedir des del host de l'aula realitzant una consulta ldapsearch dels dn.

Engegar al AWS EC2 el container ldapserver:

```
[fedora@ip-172-31-20-75 ~]$  docker run --rm --name ldap  -h ldap  --net mynet -p 389:389 -d
edtasixm06/ldapserver:18group
accc8a0c2677ffb5232a20870dcb1d82782490be1fc9c5df9c97de61a7a9f2c8

[fedora@ip-172-31-20-75 ~]$ docker ps
CONTAINER ID      IMAGE                          COMMAND                   CREATED           STATUS
PORTS                                NAMES
accc8a0c2677      edtasixm06/ldapserver:18group   "/opt/docker/startup…"   3 seconds ago      Up 2 seconds
0.0.0.0:389->389/tcp                 ldap
```

Establir el túnel ssh interactiu des del host de l'aula:

```
$ ssh -i .ssh/MyKeyPair.pem  -L 50000:localhost:389 fedora@35.178.81.76
Last login: Thu Mar  7 17:02:48 2019 from 2.137.66.200
[fedora@ip-172-31-20-75 ~]$
```

Test de la connexió segura, des del host de l'aula:

```
$ ldapsearch -x -LLL -h localhost:50000 dn
dn: dc=edt,dc=org
dn: ou=maquines,dc=edt,dc=org
dn: ou=clients,dc=edt,dc=org
...
```

Exemple-3 TD-HL-HD. Consulta postgres insegura usant un túnel ssh segur

Infraestructura: el host de l'aula (host-local) i un container docker postgres (edtasixm06/postgres:detach) en un host AWS EC2 amb el port 5432 mapejat. Obrir el firewall de AWS EC2 el port SSH i el port postgres. Accedir des del host de l'aula realitzant una consulta psql.

Engegar al AWS EC2 el container postgres:

```
<deprecated>
pendent: generar el postgres detach que al dockerfile usant user faci tota la instal·lació i el deixi en background edtasixm06/postgres:latest
        $ docker run --rm --name psql -h psql -p 5432:5432  -it edtasixm06/postgres /bin/bash
        $inicialitzar des de dins del docker
        $ psql -qtA -F',' -h d02 -U edtasixm06 training -c "select * from oficinas;"
```

nota: crec que ara ja hi ha un postgres detach que engega automàticament training

```
$ docker run --rm --name psql -h psql -p 5432:5432  -d edtasixm06/postgres:detach
$ psql -qtA -F',' -h d02 -U edtasixm06 training -c "select * from oficinas;"
```

Establir el túnel ssh interactiu des del host de l'aula:

```
$ ssh -i .ssh/MyKeyPair.pem -L 50000:localhost:5432 fedora@35.178.81.76
Last login: Thu Mar  7 17:02:48 2019 from 2.137.66.200
[fedora@ip-172-31-20-75 ~]$
```

Test de la connexió segura, des del host de l'aula:

```
$  <pendent>
```

Servei destí aïllat (localhost)

Exemple d'accedir a un servei destí que està aïllat, només ofereix servei al localhost del propi host destí. Podem practicar engegant un servei destí que faci el ***bind*** únicament a l'adreça del localhost i no a totes les seves adreces IP. En un cas així des d'un host extern (host de l'aula) no es pot accedir al servei destí.

Exemples amb:

- nethost
- Python-IPC

Exemple-4 TD-HL-HD. Nethost amb serveis aïllats al localhost

Infraestructura: el host de l'aula (host-local) en un host AWS EC2 (AMI és el host-destí) hi despleguem un i un container docker nethost (edtasixm11/net18:nethost) que ha de tenir oberts els ports 7, 13 i 80 mapejats a la AMI (fins i tot podeu fer ell map només al localhost de la AMI si recordeu com es fa amb docker). El host AWS EC2 només té obert el port 22 al firewall. Des de l'exterior no hi ha accés als serveis que ofereix la AMI (tant si estan mapejats a totes les interfícies com si ho estan només al localhost).

Engegar el nethost amb els ports mapejats en el host AWS EC2

```
[fedora@ip-172-31-20-75 ~]$ docker run --rm --name net1 -h net1  --net mynet -p 7:7 -p 13:13 -p 80:80 -d
edtasixm11/net18:nethost
bf8bdec2b6775c6af11a86b8b5a0d41a08cc0be008ae6421a918822e1ec86128

[fedora@ip-172-31-20-75 ~]$ docker ps
CONTAINER ID      IMAGE                 COMMAND                       CREATED             STATUS              PORTS
              NAMES
bf8bdec2b677        edtasixm11/net18:nethost   "/opt/docker/startup…"   About a minute ago   Up About a minute
0.0.0.0:7->7/tcp, 0.0.0.0:13->13/tcp, 0.0.0.0:80->80/tcp   net1
```

Exteriorment només s'accedeix via ssh

```
$ nmap 35.178.81.76
PORT   STATE SERVICE
22/tcp open  ssh
```

```
[fedora@ip-172-31-20-75 ~]$ nmap localhost
PORT   STATE SERVICE
7/tcp  open  echo
13/tcp open  daytime
22/tcp open  ssh
80/tcp open  http
```

Establir el túnel interactiu al host de AWS EC2.

```
$ ssh -i .ssh/MyKeyPair.pem  -L 50000:localhost:13 fedora@35.178.81.76
Last login: Thu Mar  7 18:09:48 2019 from 2.137.66.200
[fedora@ip-172-31-20-75 ~]$
```

Test des del host de l'aula:

```
$ telnet localhost 50000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
07 MAR 2019 18:20:52 UTC
Connection closed by foreign host.
```

Proveu de repetir l'exercici desplegant el container Docker amb els ports mapejats únicament al localhost de la AMI i no a totes les interfícies de la AMI.

Servei remot a un host-remot de la xarxa destí

Poseu atenció als significats diferents de destí i remot.

- ***destí***: host al que connectem per ssh, que és la porta de entrada a una xarxa destí, per exemple una màquina AMI de AWS EC2.
- ***remot***: un dels hosts que hi ha a la xarxa destí. Els hosts remots no són directament accessibles de de l'exterior. Per exemple els containers Docker que s'executen en una xarxa privada mynet dins de la màquina AMI (host-destí).
- ***host local***: un host extern que preten connectar via ssh al host-destí per accedir a un servei que ofereix el host-remot (al qual no té accés). Per exemple el host de l'aula.

Exemple d'accedir a serveis que ofereixen servidors remots. Aquests servidors estan en una xarxa destí que s'hi accedeix connectant per ssh al host destí. Aquí usarem el host de l'aula com a host-local, el host de AWS EC2 com a host-destí que fa de frontera amb una xarxa, mynet, on hi tenim tres hosts-remots corresponents a tres containers docker net1, net2 i net3 de tipus nethost.

Exemples amb:

- nethost
- ldapserver

Exemple-5 TD-HL-HR. Xarxa destí amb tres hosts remots

Aquí usarem el host de l'aula com a host-local, el host de AWS EC2 com a host destí que fa de frontera amb una xarxa, mynet, on hi tenim tres hosts remots corresponents a tres containers docker net1, net2 i net3 de tipus nethost (edtasixm11/net18:nethost). El firewall de AWS EC2 només ha de tenir obert el port 22, sense permetre cap altre accés.

Com que els hosts remots que utilitzem són containers docker cal configurar el /etc/hosts del host AWS EC2 per poder accedir als containers per nom de host (assegureu-vos de crear els containers assignant-los noms de host i de container individuals).

Engegar el nethost sense ports mapejats en el host AWS EC2

```
[fedora@ip-172-31-20-75 ~]$ docker run --rm --name net1 -h net1  --net mynet -d edtasixm11/net18:nethost
a694f67b2a709148f2e60ec3776aec98f37307736beb0649892f0b9fb99ba59e
[fedora@ip-172-31-20-75 ~]$ docker run --rm --name net2 -h net2  --net mynet -d edtasixm11/net18:nethost
64ebbc4274c9e44751ac0e57b9b73605877195eef30b0b69021e5cbd6d0cc961
[fedora@ip-172-31-20-75 ~]$ docker run --rm --name net3 -h net3  --net mynet -d edtasixm11/net18:nethost
ca031ac89140815c0227d430fba0c3ad94cd6506f32af6ba5788a5bd4c432ed1

[fedora@ip-172-31-20-75 ~]$ docker ps
CONTAINER ID   IMAGE                       COMMAND                  CREATED          STATUS          PORTS
          NAMES
ca031ac89140   edtasixm11/net18:nethost    "/opt/docker/startup…"   4 seconds ago    Up 3 seconds
net3
64ebbc4274c9   edtasixm11/net18:nethost    "/opt/docker/startup…"   14 seconds ago   Up 13 seconds
net2
a694f67b2a70   edtasixm11/net18:nethost    "/opt/docker/startup…"   24 seconds ago   Up 23 seconds
net1
```

Configurar al host AWS EC2 el hostname dels containers

```
$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
172.20.0.2 net1
172.20.0.3 net2
172.20.0.4 net3
```

Establir el túnel interactiu al host de AWS EC2

```
$ ssh -i .ssh/MyKeyPair.pem  -L 50000:net1:13 fedora@35.178.81.76
Last login: Thu Mar  7 18:16:54 2019 from 2.137.66.200
[fedora@ip-172-31-20-75 ~]$
```

Test des del host de l'aula:

```
$ telnet localhost 50000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
07 MAR 2019 18:37:30 UTC
Connection closed by foreign host.
```

Test per a incrèduls:

Si volem verificar que realment accedim al host remot indicat podem per exemple personalitzar el missatge del servidor https del container net3. Establir el túnel ssh i llavors accedir al port 50000 localment via telnet o amb un navegador.

```
[fedora@ip-172-31-20-75 ~] $ docker exec -it net3 /bin/bash

[root@net3 docker]# echo  "Test per a incrèduls, soc al NET3" > /var/www/html/index.html

[root@net3 docker]# cat /var/www/html/index.html
Test per a incrèduls, soc al NET3
```

```
$ ssh -i .ssh/MyKeyPair.pem  -L 50000:net3:80 fedora@35.178.81.76
Last login: Thu Mar  7 18:37:05 2019 from 2.137.66.200
[fedora@ip-172-31-20-75 ~]$
```

```
browser (mozilla, chrome, chromium…) llocalhost:50000
```

Exemple-6 TD-HL-HR. Servei remot ldap

Aquí usarem el host de l'aula com a host-local, el host de AWS EC2 com a host destí que fa de frontera amb una xarxa, mynet, on hi tenim un host remot ldap corresponent a un container ldapserver (edtasixm06/ldapserver:18group). El firewall de AWS EC2 només ha de tenir obert el port 22, sense permetre cap altre accés.

Com que el host remot que utilitzem és un container docker cal configurar el /etc/hosts del host AWS EC2 per poder accedir al container per nom de host.

Engegar el nethost sense ports mapejats en el host AWS EC2

```
[fedora@ip-172-31-20-75 ~]$ docker run --rm --name ldap -h ldap --net mynet -d edtasixm06/ldapserver:18group
ccb35963baba06fe5937b2bc1bdf632a0d0b34678640e8267421bf1c3378680a

[fedora@ip-172-31-20-75 ~]$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS
PORTS          NAMES
ccb35963baba   edtasixm06/ldapserver:18group  "/opt/docker/startup…"   6 seconds ago    Up 5 seconds
389/tcp        ldap
```

Configurar al host AWS EC2 el hostname dels containers

```
[fedora@ip-172-31-20-75 ~]$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
172.20.0.2 ldap
```

Establir el túnel interactiu al host de AWS EC2

```
$ ssh -i .ssh/MyKeyPair.pem  -L 50000:ldap:389 fedora@35.178.81.76
Last login: Thu Mar  7 18:43:00 2019 from 2.137.66.200
[fedora@ip-172-31-20-75 ~]$
```

Test des del host de l'aula:

```
$ ldapsearch -x -LLL -h localhost:50000 dn
dn: dc=edt,dc=org
dn: ou=maquines,dc=edt,dc=org
dn: ou=clients,dc=edt,dc=org
...
```

Visualitzacions externes

Un ús molt útil dels túnels ssh a serveis remots és visualitzar localment informació a través d'un port local que connecta a un port remot proveïdor d'informació. Per exemple des de casa volem monitoritzar gràficament un servei que està en una xarxa remota (dins de l'escola) i al que no podem accedir. Només podem accedir al host destí (host/router de l'escola), que fa de frontera entre el nostre host exterior i la xarxa remota de l'escola.

Exemple-7 TD-HL-HR. Visualització local del servei remot ldap

Aquí usarem el host de l'aula com a host-local, el host de AWS EC2 com a host destí que fa de frontera amb una xarxa, mynet, on hi tenim un host remot ldap corresponent a un container ldapserver (edtasixm06/ldapserver:18group) i un container on s'executa php (edtasixm06/hpldapadmin). El firewall de AWS EC2 només ha de tenir obert el port 22, sense permetre cap altre accés.

Pretenem visualitzar la base de dades ldap usant phpldapadmin, que tenim com a container prefabricat amb el nom: edtasixm06/phpldapadmin:latest. Aquest container s'executa també al host AWS EC2 i accedeix al servidor ldap.edt.org. Es pot accedir gràficament a l'administració de phpldapadmin usant el port 80. Però externament (des de l'aula) no podem accedir al port 80 d'un host remot (dins de la xarxa mynet del AMI).

El que farem és en el client extern (el host de l'aula) configurar un túnel ssh per accedir al port 80 del host remot phpldapadmin usant el port local 50000. Llavors amb un navegador i accedint al port 50000 podrem administrar l'ldap.

Com que els host remot que utilitzem és un container docker cal configurar el /etc/hosts del host AWS EC2 per poder accedir al container per nom de host. Cal que el host-destí (la màquina AMI identifiqui per nom de host phpldapadmin el container host-remot).

Engegar el ldap.edt.org sense ports mapejats en el host AWS EC2

```
[fedora@ip-172-31-20-75 ~]$ docker run --rm --name ldap -h ldap --net mynet -d edtasixm06/ldapserver:18group
ccb35963baba06fe5937b2bc1bdf632a0d0b34678640e8267421bf1c3378680a

[fedora@ip-172-31-20-75 ~]$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS
PORTS          NAMES
ccb35963baba   edtasixm06/ldapserver:18group  "/opt/docker/startup…"   6 seconds ago    Up 5 seconds
389/tcp        ldap
```

Repasseu que el servidor phpldapadmin estigui correctament configurat per accedir al servidor ldap anomenat *ldap.edt.org*, potser està configurat amb una adreça ip *hardcoded* o amb el nom simple *ldap* (verifiqueu quin servdor ldap useu i quin nom li heu donat).

Per refrescar la memòria sobre la utilització de phpldapadmin podeu consultar el HowTo-ASIX-LDAP.pdf dels apunts del mòdul M06-aSO de ASIX-M06.

Engegar el phpldapadmin

```
[fedora@ip-172-31-20-75 ~]$ docker run --name php -h php --net mynet -it edtasixm06/phpldapadmin /bin/bash
[root@php /]#

[fedora@ip-172-31-20-75 net18]$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS
PORTS          NAMES
e6f47dbd8b24   edtasixm06/ldapserver:18group  "/opt/docker/startup…"   35 seconds ago   Up 34 seconds
389/tcp        ldap
4d38ab4e6399   edtasixm06/phpldapadmin        "/bin/bash"              12 minutes ago   Up 12 minutes
php
```

Configuració del servei

```
$ vi /etc/phpldapadmin/config.php
** posar en lloc de la ip 172.17… el nom ldap o ldap.edt.org
/usr/sbin/httpd
/usr/sbin/httpd -S
```

Configurar el nom php per accedir remotament

```
[fedora@ip-172-31-20-75 net18]$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
172.20.0.3 php
```

Establir el túnel interactiu al host de AWS EC2

```
$ ssh -i .ssh/MyKeyPair.pem  -L 50000:php:80 fedora@35.178.81.76
Last login: Thu Mar  7 18:57:27 2019 from 2.137.66.200
[fedora@ip-172-31-20-75 ~]$
```

Visualització

```
http://localhost:50000/phpldapadmin/
```

Exemple-8 TD-HL-HL. Visualització d'un swarm remot amb visualizer ldap

Aquí usarem el host de l'aula com a host-local, el host de AWS EC2 com a host-destí que fa de frontera amb una xarxa on hem desplegat un Swarm. Aquest swarm consta d'un sol node i despleguem un container ldap i un container visualizer. El firewall de AWS EC2 només ha de tenir obert el port 22, sense permetre cap altre accés.

Fitxers de docker-compose a desplegar

```
version: "3"
services:
  ldap:
        image: edtasixm06/ldapserver:18group
        container_name: ldap.edt.org
        hostname: ldap.edt.org
        ports:
        - "389:389"
        networks:
        - mynet
  visualizer:
        image: dockersamples/visualizer:stable
        ports:
        - "8080:8080"
        volumes:
        - "/var/run/docker.sock:/var/run/docker.sock"
        deploy:
        placement:
        constraints: [node.role == manager]
        networks:
        - mynet
networks:
  mynet:
```

Engegar el swarm al host AWS EC2

```
[fedora@ip-172-31-20-75 net18]$ docker stack deploy -c docker-compose.yml myapp
```

Establir el túnel interactiu al host de AWS EC2

```
$ ssh -i ~/ecanet/.ssh/MyKeyPair.pem  -L 50000:172.31.20.75:8080 fedora@35.178.169.238
Last login: Fri Mar  8 07:56:39 2019 from 88.23.254.181
[fedora@ip-172-31-20-75 ~]$
```

Test des del host de l'aula usant un browser

```
browser localhost:50000
```

Podriem convertir l'exemple en un de host-local to host-remot si configureu al /etc/hosts de la AMI un nom com per exemple "visualizer" per accedir al port 8000 del container Docker visualize. Llavors s'accedeix directament al port del container i no al mapejat a la AMI. Consulteu l'exemple-10.

Exemple-9 TD-HL-HL. Visualització d'administració docker remota: portainer (local)

Aquí usarem el host de l'aula com a host-local, el host de AWS EC2 com a host-destí que fa de frontera amb una xarxa on hem desplegat una aplicació docker amb docker-compose. Despleguem un container ldap i un container portainer que permet l'administració remota de docker. El firewall de AWS EC2 només ha de tenir obert el port 22, sense permetre cap altre accés.

En aquest exemple el docker-compose mapeja el port 9000 al host de AWS EC2, però aquest port no és accessible des del exterior per el firewall de AWS EC2.

Fitxers de docker-compose a desplegar

```
version: "2"
services:
  nethost:
        image: edtasixm11/net18:nethost
        container_name: nethost
        hostname: nethost
        networks:
        - mynet
  portainer:
        image: portainer/portainer
        ports:
        - "9000:9000"
        volumes:
        - "/var/run/docker.sock:/var/run/docker.sock"
        networks:
        - mynet
networks:
  mynet:
```

Engegar el swarm al host AWS EC2

```
[fedora@ip-172-31-20-75 compose]$ docker-compose up -d
Creating network "compose_mynet" with the default driver
Creating nethost ...
Creating compose_portainer_1 ...
Creating nethost
Creating compose_portainer_1 ... done

[fedora@ip-172-31-20-75 compose]$ docker ps
CONTAINER ID      IMAGE                  COMMAND                  CREATED          STATUS              PORTS
        NAMES
b74e5ec8558e      portainer/portainer    "/portainer"             15 seconds ago   Up 14 seconds
0.0.0.0:9000->9000/tcp   compose_portainer_1
1290532762d7      edtasixm11/net18:nethost   "/opt/docker/startup…"   15 seconds ago   Up 14 seconds
        nethost
```

Establir el túnel interactiu al host de AWS EC2

```
$ ssh -i ~/ecanet/.ssh/MyKeyPair.pem  -L 50000:localhost:9000 fedora@35.178.169.238
Last login: Fri Mar  8 08:07:09 2019 from 88.23.254.181
[fedora@ip-172-31-20-75 ~]$
```

Test des del host de l'aula usant un browser

```
browser localhost:50000
```

Podriem convertir l'exemple en un de host-local to host-remot si configureu al /etc/hosts de la AMI un nom com per exemple "portainer" per accedir al port 9000 del container Docker portainer. Llavors s'accedeix directament al port del container i no al mapejat a la AMI. Consulteu l'exemple-10.

Exemple-10 TD-HL-HR. Visualització d'administració docker remota: portainer

Aquí usarem el host de l'aula com a host-local, el host de AWS EC2 com a host-destí que fa de frontera amb una xarxa on hem desplegat una aplicació docker amb docker-compose. Despleguem un container ldap i un container portainer que permet l'administració remota de docker. El firewall de AWS EC2 només ha de tenir obert el port 22, sense permetre cap altre accés.

En aquest exemple el docker-compose no exporta cap port al host AWS EC2 de manera que cal fer una redirecció al container portainer. Si es fa per nom caldrà configurar el /etc/hosts, però també es pot fer per l'adreça IP del container.

Fitxers de docker-compose a desplegar

```
version: "2"
services:
  ldap:
        image: edtasixm11/net18:nethost
        container_name: nethost
        hostname: nethost
        networks:
        - mynet
  portainer:
        image: portainer/portainer
        volumes:
        - "/var/run/docker.sock:/var/run/docker.sock"
        networks:
        - mynet
networks:
  mynet:
```

Engegar el swarm al host AWS EC2

```
[fedora@ip-172-31-20-75 compose]$ docker-compose up -d
Creating network "compose_mynet" with the default driver
Creating nethost ...
Creating compose_portainer_1 ...
Creating nethost
```

```
Creating compose_portainer_1 ... done

fedora@ip-172-31-20-75 compose]$ docker ps
CONTAINER ID        IMAGE                       COMMAND                  CREATED             STATUS              PORTS
                    NAMES
703b375360e4        portainer/portainer         "/portainer"             57 seconds ago      Up 55 seconds       9000/tcp
                    compose_portainer_1
1f8bfdf5b0f8        edtasixm11/net18:nethost    "/opt/docker/startup…"   57 seconds ago      Up 56 seconds
nethost
```

Esbrinar l'adreça ip del container per accedir-hi per adreça IP o per definir el nom al /etc/hosts del host AWS EC2.

```
$ docker container inspect compose_portainer_1
```

Establir el túnel interactiu al host de AWS EC2

```
$ ssh -i ~/ecanet/.ssh/MyKeyPair.pem  -L 50000:172.19.0.3:9000 fedora@35.178.169.238
Last login: Fri Mar  8 08:32:01 2019 from 88.23.254.181
[fedora@ip-172-31-20-75 ~]$
```

Test des del host de l'aula usant un browser

```
browser localhost:50000
```

Si ho preferiu podeu posra una entrada al /etc/hosts de la màquina AMI assignant un nom com per exemple "portainer" a l'adreça IP del container Docker del portainer. Així en l'ordre ssh que fa el host-local podem usar el nom de host en lloc de l'adreça ip.

Túnel SSH revers:

En aquest apartat practicarem com:

- ❏ Deixar una porta oberta de tornada.
- ❏ Obrir la porta per accedir a un altre host
- ❏ The crazy thing!

Deixar una porta oberta de tornada

En aquests exemples es tracta de deixar una porta de tornada, via ssh per accedir al host-local des de dins d'una organització. Si tenim per exemple un host-local, el de casa i no podem fer ssh a dins d'un host de l'escola, però si que des de dins de l'escola podem fer ssh al host de casa. Podem deixar des de dins de l'escola una porta oberta al host de casa per permetre l'accés, des de casa a dins de l'escola passant el firewall de l'escola.

Un exemple més aplicacable és deixar al host AMI de AWS EC2 una porta oberta per accedir al host de l'aula. Aquí el host de l'aula juga el rol de host-local des d'on farem un túnel invers connectant per ssh a la AMI de AWS EC2 i deixant-hi allà oberta la porta de tornada.

Exemple-11 TR-HL-HD. Obrir una porta de tornada en un host destí frontera

En aquest exemple tenim un host-local, el de l'aula i una organització remota, la de AWS EC2. Des de l'aula podem accedir via ssh al host de AWS EC2 que fa de frontera, que té el port ssh obert. Des d'aquest host de AWS EC2 no podem accedir al host-local, el de l'aula, perquè el firewall de l'escola no ho permet .

El que farem és obrir un túnel invers, des del host de l'escola, que deixa un port obert al host AWS EC2. D'aquesta manera des del AWS EC2 podem accedir al servei ssh del host de l'escola i connectar-nos a una sessió al nostre host-local.

Establir el túnel interactiu al host de AWS EC2

```
$ ssh -i ~/ecanet/.ssh/MyKeyPair.pem  -R 50000:localhost:22 fedora@35.178.169.238
Last login: Fri Mar  8 08:45:20 2019 from 88.23.254.181
[fedora@ip-172-31-20-75 ~]$
```

Observem que ara al AWS EC2 hi ha el port 50000 obert

```
[fedora@ip-172-31-20-75 ~]$ nmap localhost
PORT      STATE SERVICE
22/tcp    open  ssh
50000/tcp open  ibm-db2
```

Des del host AWS EC2 verificar l'accés al host de l'aula

```
[fedora@ip-172-31-20-75 ~]$ ssh -p 50000 ecanet@localhost
ecanet@localhost's password:
Last login: Fri Mar  8 08:21:31 2019
[ecanet@d02 ~]$
```

Exemple-12 TR-HL-HD. Obrir una porta de tornada a un servei

En un host de l'escola (host-local) hi despleguem serveis engegant un container nethost que mapegi almenys els ports 7, 13 i 80. Pretenem accedir des del host-destí AWS EC2 a un d'aquests serveis.

El que farem és obrir un túnel invers, des del host de l'escola (host-local), que deixa un port obert al host AWS EC2 (host-destí). D'aquesta manera des del AWS EC2 podem accedir al servei 13 del host de l'escola.

És un exemple com l'anterior però en lloc de accedint al servei ssh accedint a un altre servei.

Desplegar en el host de l'escola un container nethost amb ports mapejats

```
$ docker run --rm --name net0 -h net0 --net mynet -p 7:7 -p 13:13 -p 80:80 -d edtasixm11/net18:nethost
6321fed291f10f0e894f1189bc5ebb4335f28f03d0f6ac22ef27a1b7be1bf2ea
```

```
$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS          PORTS
          NAMES
6321fed291f1   edtasixm11/net18:nethost   "/opt/docker/startup…"   11 seconds ago   Up 9 seconds
0.0.0.0:7->7/tcp, 0.0.0.0:13->13/tcp, 0.0.0.0:80->80/tcp   net0
```

Establir el túnel interactiu al host de AWS EC2

```
$ ssh -i ~/ecanet/.ssh/MyKeyPair.pem  -R 50000:localhost:13 fedora@35.178.169.238
Last login: Fri Mar  8 08:45:20 2019 from 88.23.254.181
[fedora@ip-172-31-20-75 ~]$
```

Observem que ara al AWS EC2 hi ha el port 50000 obert

```
[fedora@ip-172-31-20-75 ~]$ nmap localhost
PORT      STATE SERVICE
22/tcp    open  ssh
50000/tcp open  ibm-db2
```

Des del host AWS EC2 verificar l'accés al host de l'aula

```
[fedora@ip-172-31-20-75 ~]$ telnet localhost 50000
Trying ::1...
Connected to localhost.
Escape character is '^]'.
08 MAR 2019 09:14:44 UTC
Connection closed by foreign host.
```

Obrir la porta per accedir a un altre host

En aquests exemples es tracta de deixar una porta de tornada en el host-destí, per accedir a serveis que no estan en el host-local sinó en un altre host de la xarxa origen (ho anomenem host-remot-origen).

Tenim per exemple vàris hosts a casa però des d'aquests i no podem fer ssh a dins d'un host de l'escola. Però si que des de dins de l'escola podem fer ssh a un host de casa. Llavors podem deixar des de dins de l'escola (host-local) una porta oberta al host de casa (host-destí) per permetre l'accés a serveis que tenim desplegats en altres hosts de casa (hosts-remots-destí), passant-nos per alt el firewall.

L'exemple més pràctic a implementar és que des d'un host de l'aula (host-local) connectem a una màquina AMI de AWS EC2 (host-destí) i hi deixem allà una porta oberta per accedir a altres hosts que estan a l'aula. Per exemple en el host-local de l'aula podem desplegar containers en una xarxa mynet privada. Des del AWS EC2 aquests container no serien accessibles, però ho seran a través del 'rebot' al host-remot.origen.
De fet el 'rebot' al host-remot-origen el podem fer a qualsevol altre host de l'aula.

Exemple-13 TR-HL-HRO Obrir una porta de tornada a un servei remot-origen

En un host de l'escola (host-local) hi despleguem serveis engegant tres containers nethost que ofereixen serveis localment i són accessibles des del host de l'escola, però no des de fora. Pretenem accedir des del host-destí AWS EC2 a un d'aquests serveis dels containers que hi ha a l'interior del host de l'escola.

El que farem és obrir un túnel invers, des del host de l'escola (host-local), que deixa un port obert al host AWS EC2 (host-destí). El port no enllaça amb el host de l'escola sinó que es redirigeix (rebota) a un dels containers (el host-remot-origen). D'aquesta manera des del AWS EC2 podem accedir al servei 13 d'un dels containers.

Per poder adreçar-nos al container cal usar la seva adreça IP o definir els noms de host al /etc/hosts del host de l'aula.

Desplegar en el host de l'escola un container nethost amb ports mapejats

```
$ docker run --rm --name net1 -h net1 --net mynet -d edtasixm11/net18:nethost
efabfaaf7ce4e387bb3cbbaba04045fa885570fd06e84c92e59e56cd3da25dc8
[ecanet@d02 compose]$ docker run --rm --name net2 -h net2 --net mynet -d edtasixm11/net18:nethost
b927c83b18211e642bc14f70ea5f4f67002bdb50038aeb00a37851de40b46c8f
[ecanet@d02 compose]$ docker run --rm --name net3 -h net3 --net mynet -d edtasixm11/net18:nethost
3adf581e9d2440c0b3161cd6fb1cef874b421a3440b838660b6ed37fc594204e

[ecanet@d02 compose]$ docker ps
CONTAINER ID   IMAGE                       COMMAND                  CREATED          STATUS          PORTS
          NAMES
3adf581e9d24   edtasixm11/net18:nethost   "/opt/docker/startup…"   4 seconds ago    Up 3 seconds    net3
b927c83b1821   edtasixm11/net18:nethost   "/opt/docker/startup…"   10 seconds ago   Up 9 seconds    net2
efabfaaf7ce4   edtasixm11/net18:nethost   "/opt/docker/startup…"   17 seconds ago   Up 16 seconds   net1
```

Configurar l'accés al container per nom de host:

```
$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
172.18.0.2  net1
172.18.0.3  net3
172.18.0.4  net4
```

Establir el túnel interactiu al host de AWS EC2

```
$ ssh -i ~/ecanet/.ssh/MyKeyPair.pem  -R 50000:net1:13 fedora@35.178.169.238
Last login: Fri Mar  8 09:14:15 2019 from 88.23.254.181
[fedora@ip-172-31-20-75 ~]$
```

Des del host AWS EC2 verificar l'accés al host de l'aula

```
$ telnet localhost 50000
Trying ::1...
Connected to localhost.
Escape character is '^]'.
08 MAR 2019 09:28:31 UTC
Connection closed by foreign host.
```

Ampliació per a incrèduls

Editar a net3 el missatge del servidor web i verificar que el missatge obtingut és aquest.

```
$ docker exec -it net3 /bin/bash
[root@net3 docker]#  echo "hola increduls. soc net3" > /var/www/html/index.html
```

```
$ ssh -i ~/ecanet/.ssh/MyKeyPair.pem  -R 50000:net3:80 fedora@35.178.169.238
Last login: Fri Mar  8 09:33:32 2019 from 88.23.254.181
```

```
[fedora@ip-172-31-20-75 ~]$ telnet localhost 50000
```

Exemple-14 TR-HL-HRO Obrir una porta de tornada al servei remot-origen ldap

<underconstruction>

The crazy thing!

En aquests exemples anem a jugar amb els serveis ldap i phpldapadmin i a posar-los de tort i dle dret a veure què passa!

Exemple-15 Ldap-remot i phpldapadmin-local

Ddesplegem dins d'un container Docker (host-remot) en una AMI (host-destí) el servei ldap amb el firewall de la AMI només obrint el port 22. Localment al host de l'aula (host-local) desplegem un container amb phpldapadmin. Aquest container ha de poder accedir a les dades ldap. des del host de l'aula volem poder visualitzar el phpldapadmin.

Desplegar el servei ldap

- en el host-remot AMI AWS EC2 engegar un container ldap sense fer map dels ports.
- en la ami cal obrir únicament el port 22
- També cal configurar el /etc/hosts de la AMI per poder accedir al container ldap per nom de host (preferentment).
- verificar que des del host de l'aula (host-local) podem fer consultes ldap.

desplegar el servei phpldapadmin

- engegar en el host de l'aula (host-local) un container docker amb el servei phpldapadmin fent map del seu port 8080 al host-local (o no).
- crear el túnel directe ssh des del host de l'aula (host-local) al servei ldap (host-remot) connectant via SSH al host AMI (host-destí).
- configurar el phpldapadmin per que trobi la base de dades ldap accedint al host de l'aula al port acabat de crear amb el túnel directe ssh.
- Ara ja podem visualitzar des del host de l'aula el servei phpldapadmin, accedint al port 8080 del container phpldapadmin o al port que hem fet map del host de l'aula (si és que ho hem fet).

Exemple-16. Ldap-local i phpldapadmin-remot

Mas madera!

> Obrir localment un ldap al host. Engegar al AWS un container phpldapadmin que usa el ldap del host d el'aula. Visualitzar localment al host de l'aula el phpldapadmin del container de AWS EC2. Ahí ez nà.

Engegar ldap i phpldapadmin i que tinguin connectivitat:

- Engegar localment el servei ldap al host-local de l'aula.
- Obrir un túnel invers SSH en la AMI de AWS EC2 (host-destí) lligat al servei ldap del host-local de l'aula.
- Engegar el servei phpldapadmin en un container Docker dins de la màquina AMI. cal confiurar-lo perquè connecti al servidor ldap indicant-li la ip de la AMI i el port obert per el túnel SSH.
- *nota* atenció al binding que fa ssh dels ports dels túnels SSH (per defecte són només al localhost).

Ara cal accedir des del host de l'aula al port 8080 del phpldapadmin per visualitzar-lo. Per fer-ho cal:

- en la AMI configutat el /etc/hosts per poder accedir per nom de host (per exemple php) al port apropiat del servei phpldapadmin.
- establir un túnel directe del host de l'aula (host-local) al host-remot phpldapadmin passant pel host-destí (la AMI).
- Ara amb un navegador ja podem visualitzar localment des del host de l'aula el phpldapadmin connectant al pot directe acabat de crear.
- *nota* atenció al binding que fa ssh dels ports dels túnels SSH (per defecte són només al localhost).

Fer que el bind del túnel revers sigui a totes les ips

Recordeu que en crear un túnel SSH s'obre un port que per defecte fa bind únicament a la interfície localhost. Per poder obrir aquest port a altres interfícies calen dues passes:

- ❏ Indicar en l'ordre ssh l'adreça IP del bind. Podem indicar 0.0.0.0 per indicar totes.
- ❏ Configurar el servei sshd modificant el fitxer /etc/ssh/sshd_config i establint la directiva *GatewayPorts yes*.

Solució:

man ssh

```
-L [bind_address:]port:host:hostport
-R [bind_address:]port:host:hostport
```

By default, TCP listening sockets on the server will be bound to the loopback interface only. This may be overridden by specifying a bind_address. An empty bind_address, or the address '*', indicates that the remote socket should listen on all interfaces. Specifying a remote bind_address will only succeed if the server's **GatewayPorts** option is enabled (see sshd_config(5)).

man sshd_config(5)

GatewayPorts
Specifies whether remote hosts are allowed to connect to ports forwarded for the client. By default, sshd(8) binds remote port forwardings to the loopback address. This prevents other remote hosts from connecting to forwarded ports. GatewayPorts can be used to specify that sshd should allow remote port forwardings to bind to non-loopback addresses, thus allowing other hosts to connect. The argument may be **no** to force remote port forwardings to be available to the local host only, **yes** to force remote port forwardings to bind to the wildcard address, or **clientspecified** to allow the client to select the address to which the forwarding is bound. The default is no.

túnel invers:
```
ssh -i ~/ecanet/.ssh/MyKeyPair.pem
        -R  0.0.0.0:50000:172.18.0.2:389
        fedora@3.8.154.195
```

túnel directe:
```
ssh -i ~/ecanet/.ssh/MyKeyPair.pem
        -L 9000:172.20.0.2:80
        fedora@3.8.154.195
```

Usar xinetd per redireccionar ports

recordeu que per acabar de fer bullir l'olla a part de desplegar serveis locals i en contaieners també hem après com usar xinetd per redirigir serveis.

de la meteixa manera que podem usar els túnels SSH per 'apropar' serveis, fer apareixer com a locals serveis remots, també podem usar els redirect de xinetd. La diferència és que amb els túnels ssh el tràfic és segur.

```
service visualizer
{
  disable         = no
  type            = UNLISTED
  socket_type     = stream
  protocol        = tcp
  wait            = no
  redirect        = 192.168.99.101 8080
  bind            = 0.0.0.0
  port            = 5003
  user            = nobody
```

```
}
```