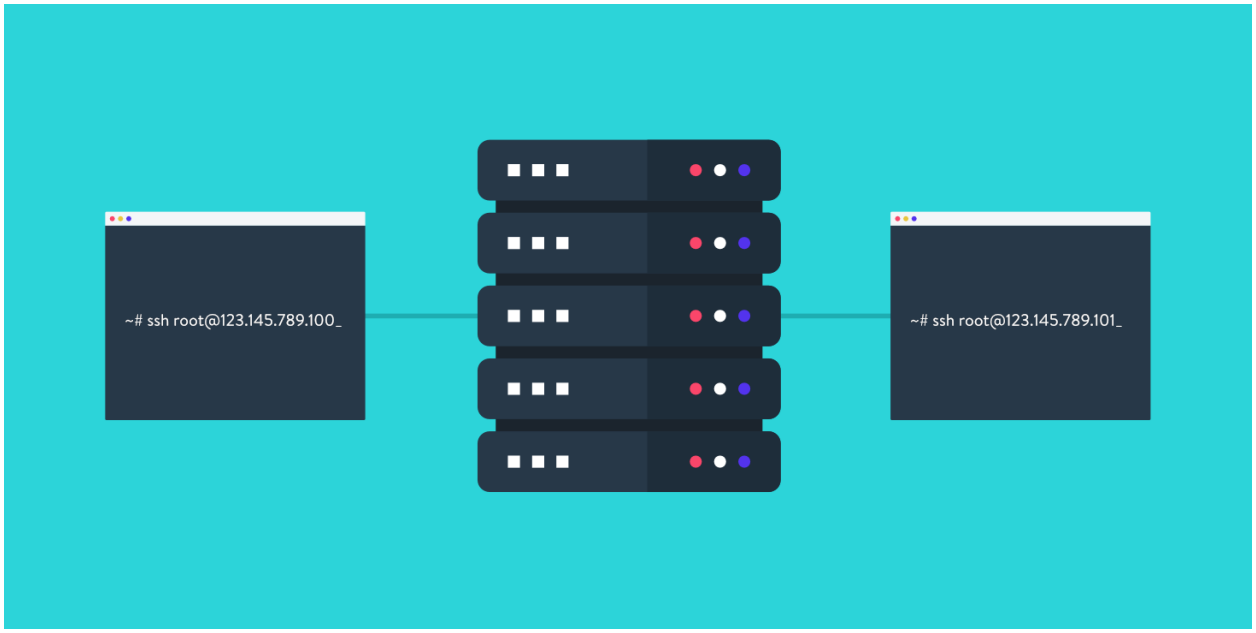


Práctica 2

Clonar la información de un sitio web

Índice

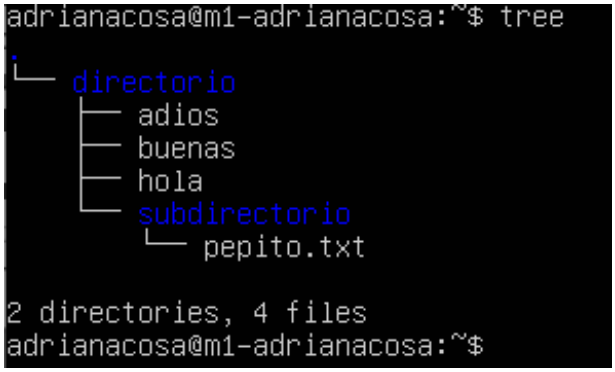
Copiar archivos locales en equipos remotos

Como se indica en el guión, debemos conocer las herramientas para poder clonar ficheros de un servidor a otro. Para ello tenemos herramientas en las que podemos pasar un directorio completo comprimido a otro servidor. Para ello podemos hacer uso de las herramientas *tar* y *scp*.

Compresión de directorios con *tar*

Imaginemos que tenemos un directorio llamado *directorio/* en nuestra máquina 1 y queremos pasarlo completamente a nuestra máquina 2. Para llevar esta tarea a cabo, en primer lugar hacemos uso de la herramienta tar, que nos permite comprimir nuestro directorio completo en un sólo fichero. En la siguiente captura vemos cómo realizamos este paso:

- Para empezar, necesitamos un directorio sobre el que trabajar. Como ya he comentado, éste se llamará *directorio*. Dentro voy a añadirle ciertos ficheros y carpetas simplemente para que sea un ejemplo de uso más práctico.



- Lo siguiente que haremos será hacer uso de la herramienta *tar* para comprimir todo el árbol de directorios que tenemos creado. En la utilización de tar vamos a usar las siguientes opciones:
    - -c, --create: sirve para crear el nuevo archivo donde se almacenará todo el contenido del directorio *directorio*.
    - -v, --verbose: nos proporciona información sobre las tareas que va realizando la ejecución de la orden tar (se ve claramente en la siguiente captura.
    - -f, --file=ARCHIVO: se utiliza para poder darle un nombre significativo al archivo que vamos a crear. Es decir, si usamos ésta opción seguida de *directorio.tar*, todo el directorio se almacenará en un fichero llamado de ésa manera.
    - -z, --gzip: se encarga de realizar una compresión con la herramienta gzip para reducir también el tamaño del archivo final.

  El resultado de la ejecución es el siguiente:

```
adrianacosa@m1-adrianacosa:~$ tar -cvzf directorio.tgz directorio/
directorio/
directorio/subdirectorio/
directorio/subdirectorio/pepito.txt
directorio/hola
directorio/adios
directorio/buenas
adrianacosa@m1-adrianacosa:~$ ls -l
total 8
drwxrwxr-x 3 adrianacosa adrianacosa 4096 Mar 15 17:28 directorio
-rw-rw-r-- 1 adrianacosa adrianacosa  230 Mar 15 17:41 directorio.tgz
adrianacosa@m1-adrianacosa:~$
```

Podemos comprobar cómo se ha reducido mucho el tamaño aún teniendo toda la estructura anteriormente comentada dentro del directorio.

Copia de archivos en servidores remotos

Ahora que ya sabemos comprimir directorios completos correctamente, procedemos a ver cómo copiar estos ficheros de un servidor a otro mediante *ssh* y *scp*. Podemos hacer la copia de estos directorios de varias maneras. Yo voy a mostrar las mencionadas en el guión y posteriormente voy a añadir algunas opciones de scp adicionales. Comenzamos con las opciones del guión:

- Como primera opción, el guión nos proporciona una orden en una sola línea en la que se hace una compresión (usando opciones anteriormente comentadas) y posteriormente mediante el uso de *ssh* creamos el fichero comprimido directamente en la máquina de destino con la siguiente orden:

  tar -czf – directorio/ | ssh usuario@equiporemoto 'cat > ~/archivo.tgz'

  Y el resultado de la ejecución de dicho comando es el siguiente:

```
adrianacosa@m1-adrianacosa:~$ tar -czf - directorio/ | ssh adrianacosa@192.168.56.102 'cat > ~/archivo.tgz'
The authenticity of host '192.168.56.102 (192.168.56.102)' can't be established.
ECDSA key fingerprint is SHA256:jszmUE+iMh8yuTrYjy4SBEF3orY/89RqhLRpL9VrRnA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.102' (ECDSA) to the list of known hosts.
adrianacosa@192.168.56.102's password:
adrianacosa@m1-adrianacosa:~$
```

```
adrianacosa@m2-adrianacosa:~$ ls -l
total 4
-rw-rw-r-- 1 adrianacosa adrianacosa 230 Mar 15 17:49 archivo.tgz
adrianacosa@m2-adrianacosa:~$
```

- La segunda opción propuesta en el guión consiste en el uso de la herramienta *scp* junto con tar. En esta opción se realiza primero la compresión en el servidor local y posteriormente se hace la copia remota del fichero generado en la compresión. Esto puede ser costoso en muchos casos en los que el directorio que estemos comprimiendo sea de un tamaño bastante grande, tanto que aunque se reduzca el tamaño con la opción -z siga siendo una opción computacionalmente costosa. Dicha opción es la siguiente:

tar -czvf archivo.tgz directorio

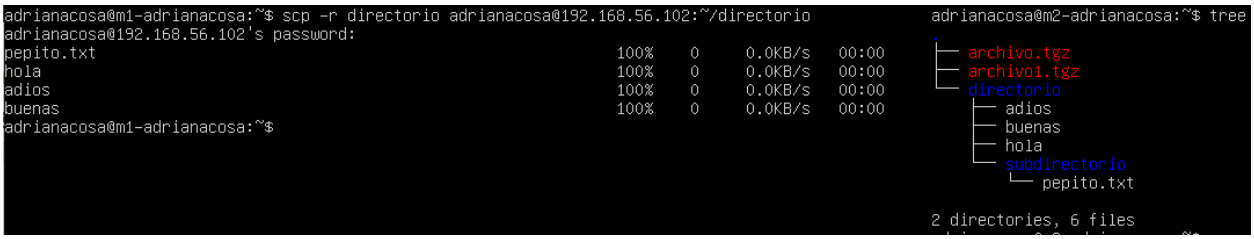
scp archivo.tgz usuario@equiporemoto:~/archivo.tgz

El resultado de la ejecución de ambos comandos resulta la siguiente:

También se propone una opción directa, en la que se copia directamente el directorio completo, pero esto también supone un inconveniente y es que, como hemos comentado antes, si el tamaño del directorio es bastante considerable, esta operación será computacionalmente más costosa que la que acabamos de ver, ya que la transferencia de los archivos es de mucho menor tamaño. La opción propuesta es la siguiente:

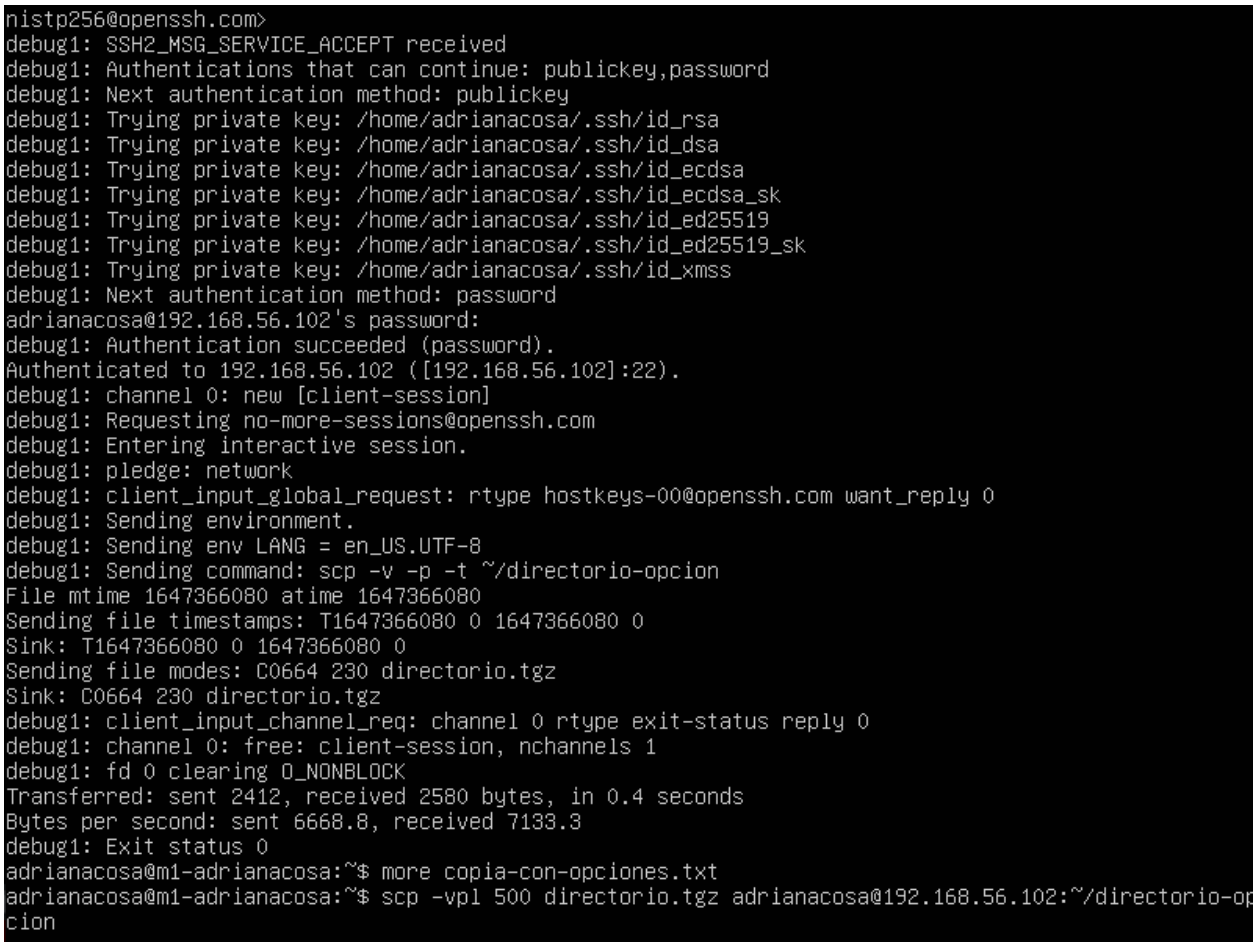
scp -r directorio usuario@equiporemoto:~/directorio

El resultado de la ejecución de este comando es el siguiente:

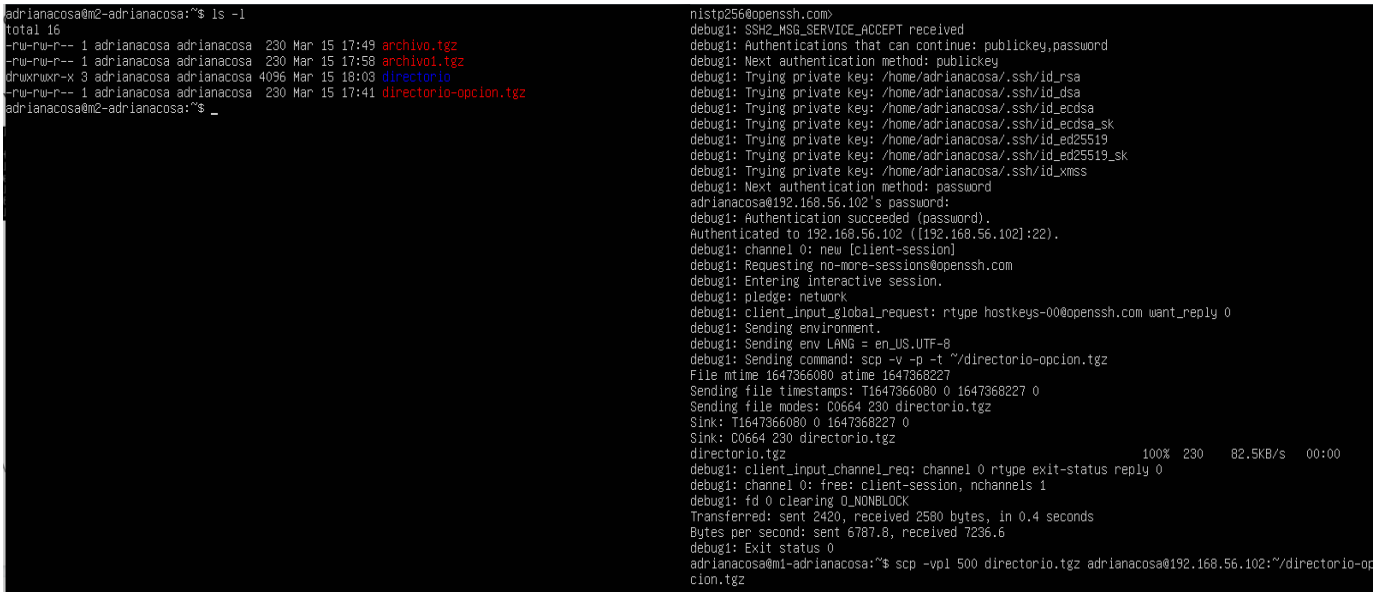


- Como uso adicional de *scp* voy a hacer una copia de un fichero haciendo uso de distintas opciones de uso de éste comando. Para ello voy a hacer uso de las opciones:
    - -l: sirve para limitar el ancho de banda usado por la orden (expresados en Kb/s).
    - -v: nos permite ver todos los mensajes sobre el progreso de la copia a modo de debug.
    - -p: mantiene los tiempos de modificación, acceso y permisos de los archivos originales

  Luego el uso de dicha orden quedaría de la siguiente manera:



Y en la máquina 2:



Rsync

La instalación del programa rsync es muy sencilla en las distintas distribuciones de Linux. Simplemente es usar el gestor de paquetes que se use en la distribución que tengamos instalada. Para nuestro caso, Ubuntu Server, el paquete ya viene instalado. En el caso de que no viniese instalado, haríamos uso del gestor de paquetes *Aptitude* e instalaríamos el paquete rsync con la orden *sudo apt-get install rsync.*

Teniendo ya disponible el paquete rsync, podemos empezar a trabajar con él. Como comenta el guión de prácticas, se puede trabajar con rsync desde el usuario sin privilegios de root para todas aquellas situaciones en las que no necesitemos unos permisos especiales sobre un directorio. Pero en este caso, si trabajamos donde Apache almacena los archivos del espacio web por defecto, vamos a tener problemas de permisos ya que se encuentran en */var/www*. Podemos solucionarlo de dos formas: o trabajamos directamente con el usuario root o damos permisos al usuario a la carpeta */var/www* para no tener ningún problema con los permisos sobre este directorio. Otra cosa a tener en cuenta es que estos permisos los tenemos que asignar en ambas máquinas, ya que el objetivo es tener sincronizar ambos servidores para tener en ambos los mismos ficheros actualizados, y para ello la máquina donde copiaremos estos archivos necesitará que el usuario que gestiona el servidor web tenga permisos también sobre ese directorio.

Teniendo en cuenta lo anterior dicho, procedemos a darle permisos a mi usuario en ambas máquinas sobre el directorio */var/www*. Para ello usaremos el comando chown de la siguiente forma:

```
adrianacosa@m1-adrianacosa:~$ sudo chown adrianacosa:adrianacosa -R /var/www
[sudo] password for adrianacosa:
adrianacosa@m1-adrianacosa:~$
```

Para comprobar que *rsync* funciona correctamente y que le hemos asignado los permisos correctamente vamos a clonar la carpeta que almacena el contenido del servidor web principal. Para ello vamos a ejecutar en la máquina 2 la siguiente orden:

```
adrianacosa@m2-adrianacosa:/var/www/html$ rsync /var/www/ -avz -e ssh 192.168.56.102:/var/www/
The authenticity of host '192.168.56.102 (192.168.56.102)' can't be established.
ECDSA key fingerprint is SHA256:G4RHIKCJIXcG7+GMqsJ/u9VkyqcZ7qTN1uS6z1TfQ3k.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.102' (ECDSA) to the list of known hosts.
adrianacosa@192.168.56.102's password:
sending incremental file list
./
html/
html/index.html
html/swap.html

sent 374 bytes  received 167 bytes  83.23 bytes/sec
total size is 11,102  speedup is 20.52
adrianacosa@m2-adrianacosa:/var/www/html$
```

Donde las opciones que le hemos indicado son las siguientes:

- -a, --archive: es una opción que activa muchas otras como la recursividad, conservar enlaces simbólicos, fechas de modificación, los permisos, transfiere la propiedad de usuario y grupo y copia dispositivos especiales.
- -v, --verbose: se activa el modo explicativo, en el que se imprime por pantalla todos los pasos que va siguiendo la ejecución del programa.
- -z, --compress: activa la compresión durante el envío de los archivos al servidor remoto.
- -e: sirve para especificar el shell remoto a utilizar.

También podemos hacer una simulación de una copia de los datos a un servidor remoto con rsync. Esta opción es la opción *--dry-run* y nos permite hacer pruebas para aprender a usar bien rsync. Por ejemplo, imaginemos que queremos simular unas condiciones concretas en las que se realizaría una sincronización de ficheros con una velocidad de red específica. Pues podríamos hacer lo que aparece en la siguiente captura:

Y por último, una opción que me parece interesante, es la de *--size-only,* que se encarga de realizar una copia sólo de los ficheros en los que ciertos metadatos tales como el tamaño o

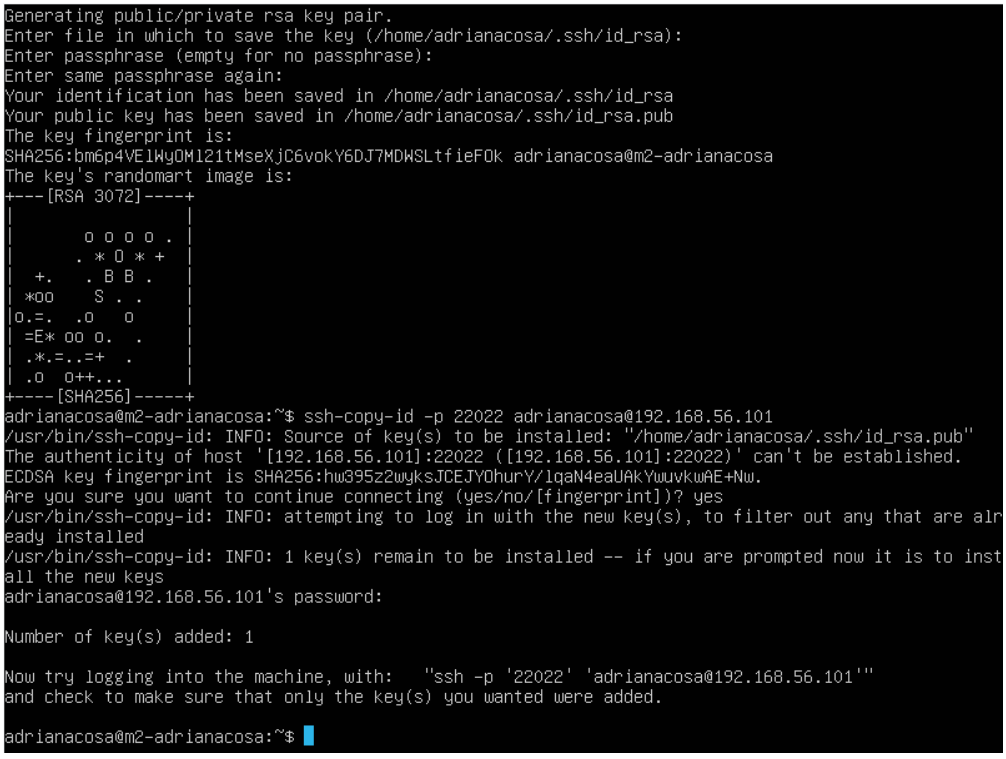
tiempo de acceso hayan cambiado con respecto a su copia en el servidor remoto. De esta forma nos limitamos sólo a copiar los ficheros justos y necesarios, ya que si no hemos modificado un fichero desde la última vez que lo sincronizamos con el servidor remoto, no hay necesidad de volver a mandarlo con la misma información. Un ejemplo de uso sería el siguiente:

Configuración de SSH para acceso sin contraseña

Esta configuración ya la hice en la práctica 1. Voy a copiar la misma información y a añadir la copia manual de la clave (en dicha configuración estaba incluido el cambio de puerto. En esta práctica no voy a hacer cambio de puerto). La parte que está en formato distinto es la de la práctica anterior:

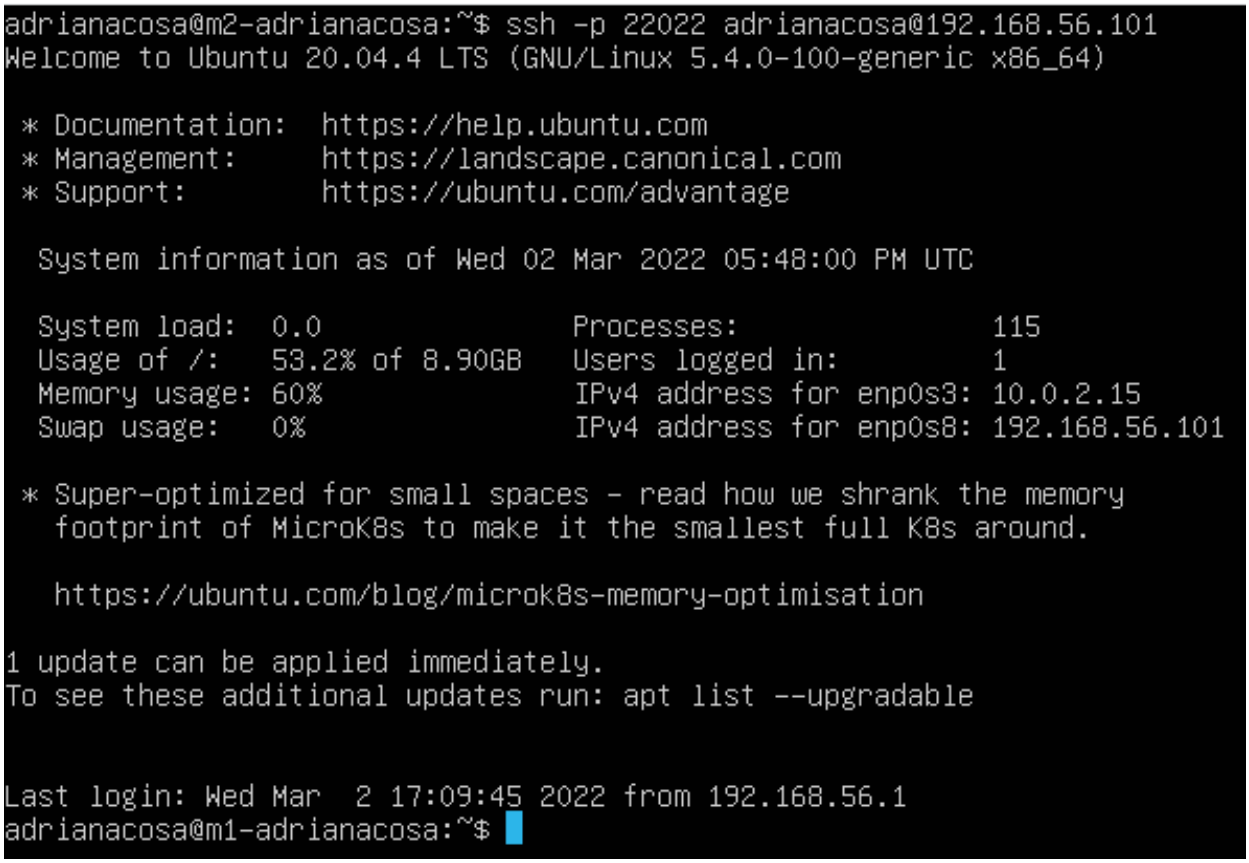
Inicio de sesión sin contraseña (clave pública)

Ahora vamos a pasar a los clientes SSH, que serán la máquina 2 y la máquina anfitrión. Lo primero que tenemos que hacer es generar una clave SSH en la máquina dos que será copiada en la máquina 1 para que quede registrada como conocida y así no necesitar contraseña para entrar. Para ello tenemos que seguir los siguientes pasos que aparecen en las imágenes siguientes (el primer comando es ssh-keygen, pero sale cortado en la máquina virtual):



Para poder copiar nuestra clave pública en el servidor SSH lo hacemos mediante el comando “ssh-copy-id”, a la que le indicamos el usuario y la IP del servidor al que queremos mandar nuestra clave pública y éste nos pedirá que iniciemos sesión para comprobar que verdaderamente somos nosotros los que le estamos mandando la clave.

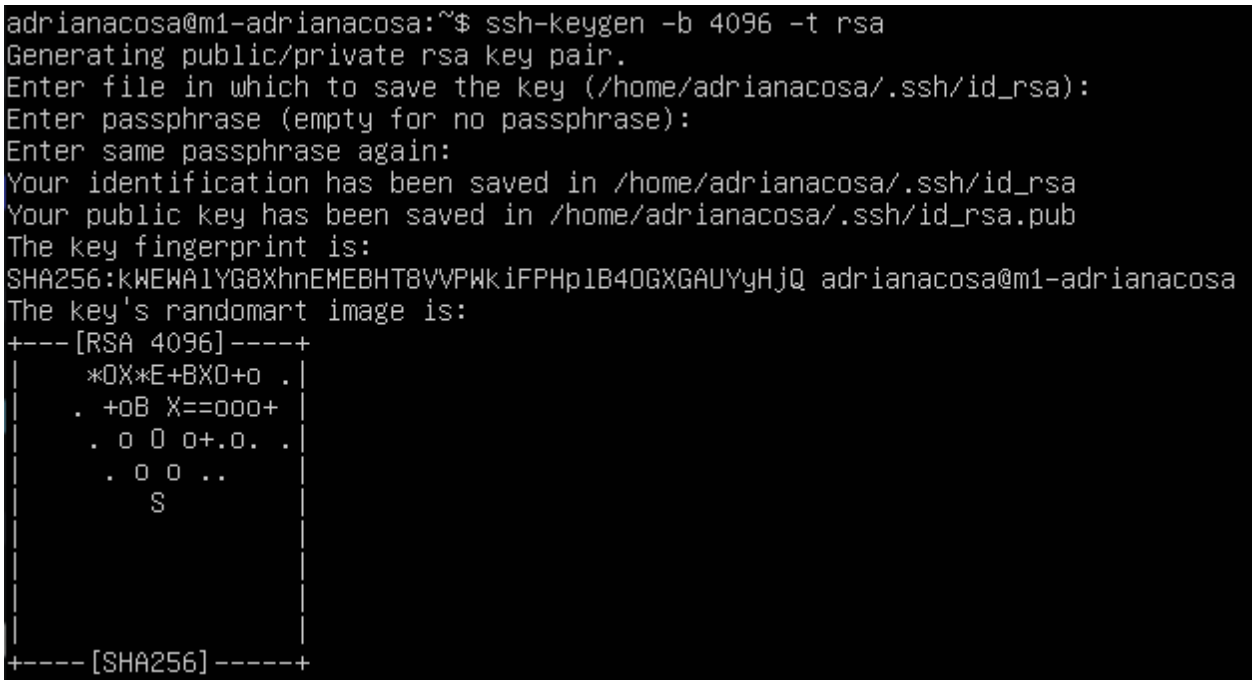
Una vez tenemos esto hecho, probamos como dice al final a conectarnos por SSH a la máquina 1 y vemos si pide o no la contraseña:



Y como vemos no la pide.

Con esto hemos terminado de configurar por completo SSH. Resumiendo: hemos configurado para SSH un puerto distinto al 22 (el que viene por defecto), hemos habilitado el acceso por clave pública y hemos habilitado la autenticación por contraseña para poder confirmar nuestra identidad a la hora de conectarnos al servidor SSH.

A partir de esto vamos a ver las opciones adicionales que se proponen en el guión. La primera opción que se nos propone es la siguiente:



- La opción *-b* nos indica el número de bits a usar en la codificación de nuestra clave. En este caso le hemos indicado que realice la codificación con 4 Kb.
- La opción *-t* sirve para especificar el algoritmo de cifrado a utilizar para generar la clave.

Para hacer la copia de la clave que acabamos de generar al servidor al que queremos acceder sin contraseña podemos hacerlo de manera muy sencilla usando el comando *ssh-copy-id*. Esto se haría de la manera que ya viene indicada en las capturas de pantalla de la práctica anterior. Iba a usar distintas opciones para ampliar lo descrito en la memoria de la práctica anterior, pero viendo las pocas opciones que existen, voy a explicar las que me parecen más interesantes:

- Existe la opción *-n* que permite simular el copiado de una clave a un servidor en concreto. El resultado de la ejecución con este comando no hace nada, simplemente nos indica la clave que se copiaría en ese caso.
- También tenemos la opción *-i* que nos permite volcar todas las claves que tengamos en un fichero hacia el servidor al que queremos acceder. Esto nos facilita mucho el trabajo a la hora de copiar múltiples claves de una sola vez.
- Por último, nombrar la opción *-f*, que obliga a copiar la clave aunque ya se encuentre en el servidor donde la copiamos. Es decir, le decimos que no realice una comprobación previa de si existe dicha clave para no volver a copiarla en ese caso, la orden copiará la clave igualmente.

Imitando el comportamiento de ssh-copy-id

Por último, podemos deducir el funcionamiento de *ssh-copy-id* según lo visto en el apartado de *scp*. Podemos intentar copiar nuestra clave pública de manera manual mediante el uso de este comando, así que vamos a probarlo. En primer lugar generamos una clave con nombre *manual.pub* en el servidor 2 para copiarlo en el 1 manualmente:

```
adrianacosa@m2-adrianacosa:~$ ssh-keygen -b 1024 -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/adrianacosa/.ssh/id_dsa): manual
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in manual
Your public key has been saved in manual.pub
The key fingerprint is:
SHA256:sESswbtSs8CRIRo/d+2Yiyv3ZiXX4PFraW8CTIrp7BI adrianacosa@m2-adrianacosa
The key's randomart image is:
+---[DSA 1024]----+
|o .+ ..          |
|.+o o...         |
|..o..++ .        |
|  oo=o *+        |
|   o *+=S=       |
|  E =.o.* o      |
|   =. .+ . o     |
|  o +.o   * .    |
|   =o+.  o +.    |
+----[SHA256]-----+
adrianacosa@m2-adrianacosa:~$ _
```

Una vez tenemos la clave, usamos *scp* para copiarla en el /home de nuestro usuario en el servidor donde queremos copiarlo (en este caso en servidor 1):

```
adrianacosa@m2-adrianacosa:~$ scp /home/adrianacosa/manual.pub adrianacosa@192.168.56.102:/home/adri
anacosa/
The authenticity of host '192.168.56.102 (192.168.56.102)' can't be established.
ECDSA key fingerprint is SHA256:G4RHIKCJIXcG7+GMqsJ/u9VkyqcZ7qTN1uS6z1TfQ3k.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.102' (ECDSA) to the list of known hosts.
adrianacosa@192.168.56.102's password:
Permission denied, please try again.
adrianacosa@192.168.56.102's password:
manual.pub                                    100%  616     1.2MB/s   00:00
adrianacosa@m2-adrianacosa:~$
```

Y lo útlimo que nos quedaría sería copiar la clave al archivo donde se almacenan todas las claves que serán reconocidas por el servidor si intentan conectarse por *ssh* y de esta manera se podrán autenticar sin contraseña:

```
adrianacosa@m1-adrianacosa:~$ cat manual.pub >> .ssh/authorized_keys
adrianacosa@m1-adrianacosa:~$ _
```

He investigado porque me seguía pidiendo la contraseña, y justo hay que añadir a la configuración de *SSH* que permita la autenticación mediante claves dsa (la usada en este ejemplo) ya que si no buscará solamente claves RSA. La línea que habría que añadir tiene la forma:

DSAAuthentication yes

Programar tareas con Crontab

El guión nos explica el funcionamiento del demonio *cron* para hacer uso de tareas programadas. Un ejemplo muy sencillo de su uso puede ser el de tener un *cold spare server* constantemente actualizado por si en algún momento se cae la conectividad del servidor de producción tengamos los mismos ficheros del servidor web que éste último.

Para ello el guión propone que programemos con *cron* la copia automática cada hora del contenido del directorio */var/www* al otro servidor. Esto se realiza añadiendo al fichero */etc/crontab* la siguiente línea:

*0 * * * * root rsync /var/www/ -avz -e ssh adrianacosa2122@192.168.56.101:/var/www/*

Lo que le indicamos a cron en esta línea es que clone la carpeta */var/www* con rsync y desde el usuario root al servidor *m2-adrianacosa* en su */var/www* tal y como he explicado en el apartado de rsync. Hay que tener en cuenta que para que este comando se ejecute correctamente, debemos poder acceder a éste mediante ssh sin contraseña.

También decir que no hay porqué entrar a editar el archivo */etc/crontab* para programar tareas con cron. Podemos añadir tareas al demonio desde la línea de comandos usando la orden *crontab*. Por ejemplo, si queremos añadir la misma orden, simplemente añadimos delante de lo escrito arriba la palabra *crontab* en el terminal y sin indicar el usuario. Es decir, habría que indicarlo así:

*0 * * * * rsync /var/www/ -avz -e ssh adrianacosa2122@192.168.56.101:/var/www/*

Existen palabras reservadas para lapsos de tiempo usados comúnmente. El ejemplo anterior podríamos haberlo programado con la palabra reservada *@hourly* en lugar de los 5 parámetros. Y existen muchas del estilo como:

- *@reboot* : ejecutar una vez después del inicio.
- *@yearly/@annually* : una vez al año.
- *@monthly* : una vez al mes.
- *@weekly* : una vez a la semana.
- *@dialy/@midnight* : una vez al día.

Ejemplo de borrado de los directorios y archivos vacíos de la carpeta */tmp* una vez cada día:

@midnight adrianacosa find /tmp -type f -empty -delete

Por último, comentar que el administrador del sistema puede programar tareas recurrentes en distintos ficheros *cron*. Éstos indican la frecuencia escritos tras un punto como sigue:

- */etc/cron.d*
- */etc/cron.daily :* para tareas diarias.
- */etc/cron.hourly :* para tareas cada hora.
- */etc/cron.monthly :* para tareas mensuales.
- */etc/cron.weekly :* para tareas semanales.

Bibliografía

1. https://linux.die.net/man/1/tar
2. https://linux.die.net/man/1/scp
3. https://javierin.com/rsync-funcionamiento-opciones/
4. https://www.linuxtotal.com.mx/index.php?cont=rsync-manual-de-uso
5. https://crontab.guru/
6. https://geekflare.com/es/crontab-linux-with-real-time-examples-and-tools/