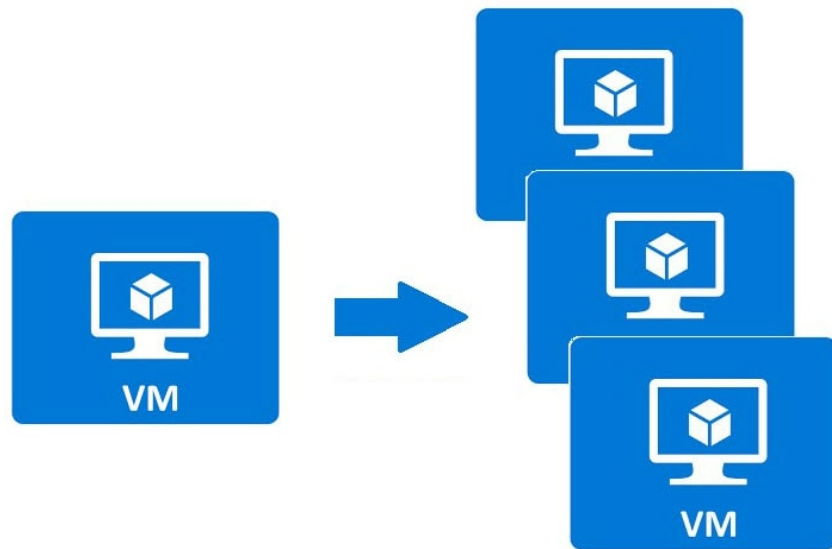


# PRÁCTICA 2: CLONAR LA INFORMACIÓN DE UN SITIO WEB

**Iñaki Melguizo Marcos**



Para la realización de la práctica he utilizado las dos máquinas virtuales m1-imm98 y m2-imm98 con idénticas características a las que tenía de la práctica anterior, ambas con UbuntuServer18.04.5.

Las direcciones IP estáticas de ambas máquinas virtuales eran:

m1-imm98	192.168.56.105
m2-imm98	192.168.56.104

y durante toda la práctica tendré configurado que el puerto de SSH de la máquina m1-imm98 será el 2222 y el de la máquina m2-imm98 el 22 (el predeterminado).

Tras esta breve introducción vamos a proceder a realizar las tareas que se piden en esta práctica.

# COPIAR ARCHIVOS LOCALES EN EQUIPOS REMOTOS

En el caso de que necesitemos crear un archivo `tgz` (en nuestro caso `archivo.tgz`) en un equipo (`m1-imm98`) y dejarlo en otro (`m2-imm98`) pero no disponemos del suficiente espacio en disco local utilizaremos el comando:

```
tar -czf diraux | ssh imm98@192.168.56.104 'cat > ~/archivo.tgz'
```

```
imm98@m1-imm98:~$ tar -czf diraux | ssh imm98@192.168.56.104 'cat > ~/archivo.tgz'
key_load_public: invalid format
tar: Cowardly refusing to create an empty archive
Try 'tar --help' or 'tar --usage' for more information.
Load key "/home/imm98/.ssh/id_rsa": invalid format
imm98@192.168.56.104's password:
```

Las opciones de `tar` que hemos usado han sido:

- `c` – crea un nuevo archivo `tgz`
- `f` – para indicarle el nombre del archivo.
- `z` – representa la compresión `zip`.

Y vemos que se nos ha creado dicho archivo `archivo.tgz` en la máquina `m2-imm98`

```
imm98@m2-imm98:~$ ls
arch2.txt  archivo.tgz
```

También podemos usar **SCP** junto con **TAR** que hace uso de **SSH**. El inconveniente del siguiente comando es que creamos el archivo comprimido `tgz` en ambas máquinas. Es decir, lo tenemos duplicado:

```
tar -czvf archivo1.tgz diraux
```

```
scp archivo1.tgz imm98@192.168.56.104:~/archivo1.tgz
```

```
imm98@m1-imm98:~$ tar -czvf archivo1.tgz diraux
diraux/
diraux/fich1.txt
diraux/dir1/
imm98@m1-imm98:~$ ls
arch1.txt  archivo1.tgz  cookies.txt  diraux  logo3w.png
imm98@m1-imm98:~$ scp archivo1.tgz imm98@192.168.56.104:~/archivo1.tgz
key_load_public: invalid format
Load key "/home/imm98/.ssh/id_rsa": invalid format
imm98@192.168.56.104's password:
archivo1.tgz                               100% 167   193.0KB/s   00:00
```

y vemos que efectivamente tenemos el archivo `archivo1.tgz` también en la máquina `m2-imm98`:

```
imm98@m2-imm98:~$ ls
arch2.txt  archivo.tgz  archivo1.tgz
```

Una opción más eficiente sería

```
scp -r diraux imm98@192.168.56.104:~/dirm2
```

que te copia el directorio diraux de la máquina m1-imm98 en el directorio dirm2 de la máquina m2-imm98:

```
imm98@m1-imm98:~$ scp -r diraux imm98@192.168.56.104:~/dirm2
fich1.txt 100% 0 0.0KB/s 00:00
```

Comprobamos el resultado:

```
imm98@m2-imm98:~$ ls
arch2.txt  archivo.tgz  archivo1.tgz  dirm2
imm98@m2-imm98:~$ cd dirm2/
imm98@m2-imm98:~/dirm2$ ls
diraux
imm98@m2-imm98:~/dirm2$ cd diraux/
imm98@m2-imm98:~/dirm2/diraux$ ls
dir1  fich1.txt
```

Otros comandos con scp serían:

- **-i**: Selecciona el archivo desde el que se lee la identidad (clave privada) para la autenticación de clave pública.
- **-P**: Le indica a ssh el puerto al que debe conectarse

En este caso tenemos la clave privada y la pública de m1-imm98 en los archivos mykey y mykey.pub

```
imm98@m1-imm98:~/.ssh$ ls
authorized_keys  id_rsa  id_rsa.pub  known_hosts  mykey  mykey.pub
```

por lo que el comando

```
scp -i ~/.ssh/mykey -P 2222 -r diraux imm98@192.168.56.104
```

copia el directorio diraux de la máquina m1-imm98 en el directorio dirm2 de la máquina m2-imm98 (a la que le hemos cambiado el puerto de ssh al 2222 para realizar esta prueba)

```
imm98@m1-imm98:~$ scp -i ~/.ssh/mykey -P 2222 -r diraux imm98@192.168.56.104:~/dirm2
fich1.txt 100% 0 0.0KB/s 00:00
```

El comando **tar -tvf** nos muestra el contenido de un archivo tgz:

```
tar -tvf archivo.tgz
```

```
imm98@m1-imm98:~$ tar -tvf archivo1.tgz
drwxrwxr-x imm98/imm98      0 2021-03-17 14:44 diraux/
-rw-rw-r-- imm98/imm98      0 2021-03-17 14:44 diraux/fich1.txt
drwxrwxr-x imm98/imm98      0 2021-03-17 14:44 diraux/dir1/
```

Para descomprimir un archivo tgz utilizaré la opción **-xvf** de tar:

`tar -xvf archivo1.tgz`

Esta opción me descomprimirá el archivo tgz que contenía el directorio 'diraux':

```
imm98@m1-imm98:~$ ls
arch1.txt  archivo1.tgz  cookies.txt  date2.txt  fecha.txt  fichd.txt  logo3w.png
imm98@m1-imm98:~$ tar -tvf archivo1.tgz
drwxrwxr-x imm98/imm98      0 2021-03-17 14:44 diraux/
-rw-rw-r-- imm98/imm98      0 2021-03-17 14:44 diraux/fich1.txt
drwxrwxr-x imm98/imm98      0 2021-03-17 14:44 diraux/dir1/
imm98@m1-imm98:~$ tar -xvf archivo1.tgz
diraux/
diraux/fich1.txt
diraux/dir1/
imm98@m1-imm98:~$ ls
arch1.txt  cookies.txt  diraux  fichd.txt
archivo1.tgz  date2.txt  fecha.txt  logo3w.png
```

## UTILIZAR RSYNC

Para utilizar la herramienta rsync, en algunas máquinas es necesario instalarlas (en las mías estaba ya instalado). Para instalarlo hay que utilizar el comando

```
sudo apt-get install rsync
```

Primeramente, vamos a hacer que el usuario sea el dueño de la carpeta donde están los archivos que hay en el espacio web (en las dos máquinas: m1-imm98 y m2-imm98):

```
sudo chown usuario:usuario -R /var/www
```

```
imm98@m1-imm98:~$ sudo chown imm98:imm98 -R /var/www
```

Vamos a proceder ya a ejecutar comandos con **rsync**. Vamos a clonar la carpeta con el contenido del servidor web principal de la máquina m2-imm98 en la máquina m1-imm98.

```
rsync -avz -e ssh 192.168.56.104:/var/www/ /var/www/
```

Vemos que el contenido de la carpeta de /var/www/html en m1-imm98 era este:

```
imm98@m1-imm98:/var/www/html$ ls
ejemplo.html  index.html
```

y el contenido de la carpeta de /var/www/html en m2-imm98 este:

```
imm98@m2-imm98:/var/www/html$ ls
ejemplo.html  ejemplovacio.html  index.html
```

Tras la ejecución de rsync tenemos:

```
imm98@m1-imm98:~$ rsync -avz -e ssh 192.168.56.104:/var/www/ /var/www/
receiving incremental file list
./
html/
html/ejemplo.html
html/ejemplovacio.html
html/index.html

sent 194 bytes  received 381 bytes  383.33 bytes/sec
total size is 11,021  speedup is 19.17
imm98@m1-imm98:~$ cd /var/www/html/
imm98@m1-imm98:/var/www/html$ ls
ejemplo.html  ejemplovacio.html  index.html
```

Ahora vamos a realizar dicha clonación de carpetas pero desde m2-imm98 a m1-imm98. Como en m1-imm98 tenemos que el puerto de ssh es 2222 en el comando rsync tenemos que poner **'ssh -p 2222'**. Por lo tanto el comando resultante sería:

```
rsync -avz -e 'ssh -p 2222' 192.168.56.105:/var/www/ /var/www/
```

```
imm98@m2-imm98:/var/www/html$ rsync -avz -e 'ssh -p 2222' 192.168.56.105:/var/www /var/www
key_load_public: invalid format
The authenticity of host '[192.168.56.105]:2222 ([192.168.56.105]:2222)' can't be established.
ECDSA key fingerprint is SHA256:KNhKE/23SQicHt2baEMHXo5XRRZE1WJvCcd28ZzeI70.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[192.168.56.105]:2222' (ECDSA) to the list of known hosts.
Load key "/home/imm98/.ssh/id_rsa": invalid format
imm98@192.168.56.105's password:
receiving incremental file list
www/
www/html/
www/html/.ejemplo.html.swp
www/html/ejemplo.html
www/html/ejemplovacio.html
www/html/index.html

sent 112 bytes  received 3,883 bytes  170.00 bytes/sec
total size is 23,309  speedup is 5.83
```

La opción **--delete** nos permite especificar qué directorios/archivos copiar y cuáles ignorar en el proceso de copia. Por ejemplo tengo los siguientes archivos en el directorio /var/www/html de m2-imm98:

```
imm98@m2-imm98:/var/www/html$ ls
ejemplo.html ejemplovacio.html index.html index2.html
```

y ese mismo directorio en la máquina m1-imm98 está vacío:

```
imm98@m1-imm98:/var/www/html$ ls
imm98@m1-imm98:/var/www/html$
```

Quiero copiar todos los directorios de la carpeta /var/www/html de m2-imm98 a la carpeta de m1-imm98 excepto el archivo /var/www/html/ejemplovacio.html. Para ello ejecuto el siguiente comando:

```
rsync -avz --delete --exclude=**/html/ejemplovacio.html -e ssh 192.168.56.104:/var/www/ /var/www/
```

y vemos que el comando se ejecuta correctamente:

```
imm98@m1-imm98:~$ rsync -avz --delete --exclude=**/html/ejemplovacio.html -e ssh 192.168.56.104:/var/www/ /var/www/
receiving incremental file list
deleting html/.ejemplo.html.swp
./
html/
html/ejemplo.html
html/index.html
html/index2.html
www/
www/html/
www/html/.ejemplo.html.swp
www/html/ejemplo.html
www/html/index.html

sent 188 bytes  received 7,277 bytes  4,976.67 bytes/sec
total size is 34,330  speedup is 4.60
imm98@m1-imm98:~$ cd /var/www/html/
imm98@m1-imm98:/var/www/html$ ls
ejemplo.html index.html index2.html
```

Si por el contrario queremos incluir un determinado archivo utilizaremos la opción **--include**. Por ejemplo si queremos incluir el archivo que antes habíamos excluido, es decir, el archivo `/var/www/html/ejemplovacio.html` tendríamos que ejecutar el comando

```
rsync -avz --delete --include=/home/imm98/ejemplovacio.html -e ssh  
192.168.56.104:/var/www/ /var/www/
```

```
imm98@m1-imm98:/var/www$ rsync -avz --delete --include=**/html/ejemplovacio.html -e ssh 192.168.56.104:/var/www/ /var/www/  
receiving incremental file list  
./  
html/ejemplovacio.html  
www/  
www/html/  
www/html/.ejemplo.html.swp  
www/html/ejemplo.html  
www/html/ejemplovacio.html  
www/html/index.html  
  
sent 166 bytes  received 4,088 bytes  2,836.00 bytes/sec  
total size is 34,330  speedup is 8.07
```

Comprobamos que se ha ejecutado correctamente:

```
imm98@m1-imm98:/var/www/html$ ls  
ejemplo.html  ejemplovacio.html  index.html  index2.html
```

Otra opción de rsync es **--remove-source-files**. La funcionalidad de esta opción es que clona las carpeta de una máquina en otra y borra el contenido en la carpeta de origen. Vamos a ejecutar el comando para que clone el contenido de la carpeta `/var/www/html` de `m2-imm98` en `m1-imm98` y borre el contenido de la carpeta en `m2-imm98`:

```
rsync -avh --remove-source-files -e ssh 192.168.56.104:/var/www/ /var/www/
```

```
imm98@m1-imm98:/var/www/html$ rsync -avh --remove-source-files -e ssh 192.168.56.104:/var/www/ /var/www/  
receiving incremental file list  
  
sent 53 bytes  received 200 bytes  168.67 bytes/sec  
total size is 11.02K  speedup is 43.56
```

y vemos que tras ejecutar esta orden la carpeta `/var/www/html` de `m2-imm98` está vacía:

```
imm98@m2-imm98:/var/www/html$ ls
```

La última opción de rsync que vamos a ver es **--min-size** que solo te clona de una carpeta a otra los archivos cuyo tamaño supere el valor de min-size. Por ejemplo si queremos que solo se copien los archivos cuyo tamaño sea superior a 10KB tendremos que escribir

```
rsync -avh --min-size=10kb -e ssh 192.168.56.104:/var/www/ /var/www/
```

Vemos que el único archivo que ocupa más de 10Kb en el directorio /var/www/html de m2-imm98 es index.html:

```
imm98@m2-imm98:/var/www/html$ ls -l
total 16
-rw-r--r-- 1 imm98 imm98  103 Mar  3 15:02 ejemplo.html
-rw-rw-r-- 1 imm98 imm98    0 Mar 17 15:30 ejemplovacio.html
-rw-r--r-- 1 imm98 imm98 10918 Mar  3 14:11 index.html
-rw-rw-r-- 1 imm98 imm98    0 Mar 17 15:44 index2.html
```

Ejecutamos el comando rsync descrito anteriormente:

```
imm98@m1-imm98:/var/www/html$ rsync -avh --min-size=10kb -e ssh 192.168.56.104:/var/www/ /v
ar/www/
receiving incremental file list
./
html/
html/index.html

sent 54 bytes  received 11.18K bytes  7.49K bytes/sec
total size is 11.02K  speedup is 0.98
```

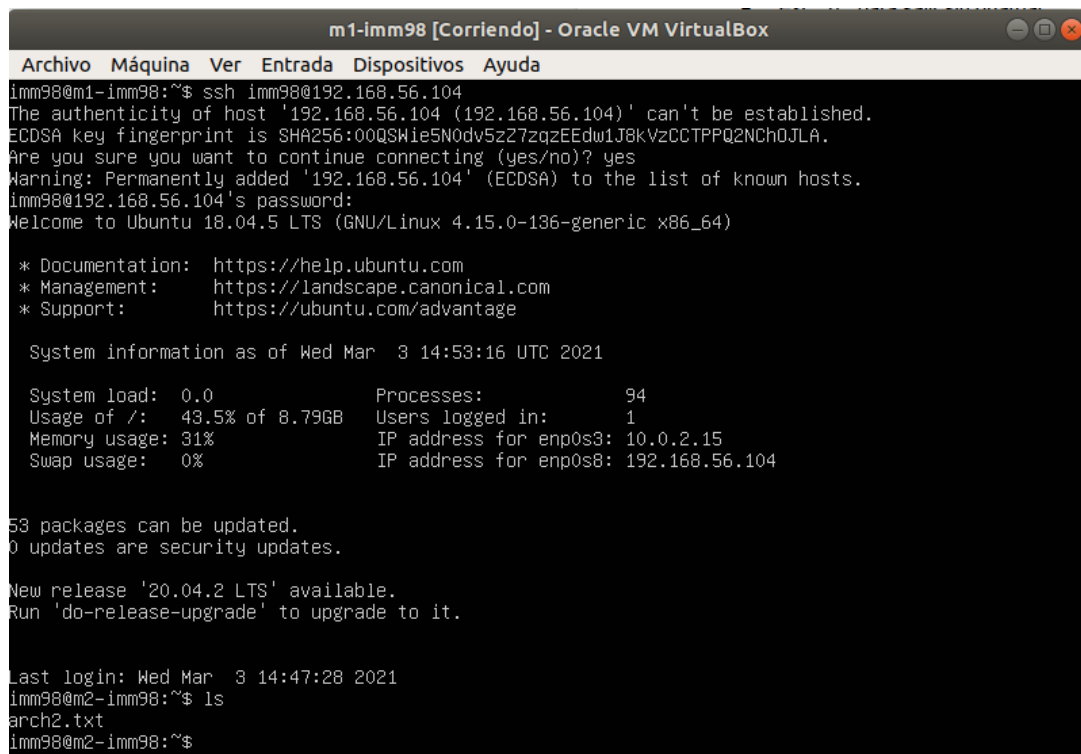
y vemos que sólo se copia el archivo index.html



# CONFIGURACIÓN DE SSH PARA EVITAR INTRODUCIR CONTRASEÑA

En la anterior práctica vimos cómo conectarnos con ssh de una máquina a otra. A modo de recordatorio:

- de m1 a m2:



```
m1-imm98 [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
imm98@m1-imm98:~$ ssh imm98@192.168.56.104
The authenticity of host '192.168.56.104 (192.168.56.104)' can't be established.
ECDSA key fingerprint is SHA256:00QSKie5N0dv5zZ7zqzEEdwIJ8kVzCCTPPQ2NCh0JLA.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.56.104' (ECDSA) to the list of known hosts.
imm98@192.168.56.104's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-136-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Wed Mar  3 14:53:16 UTC 2021

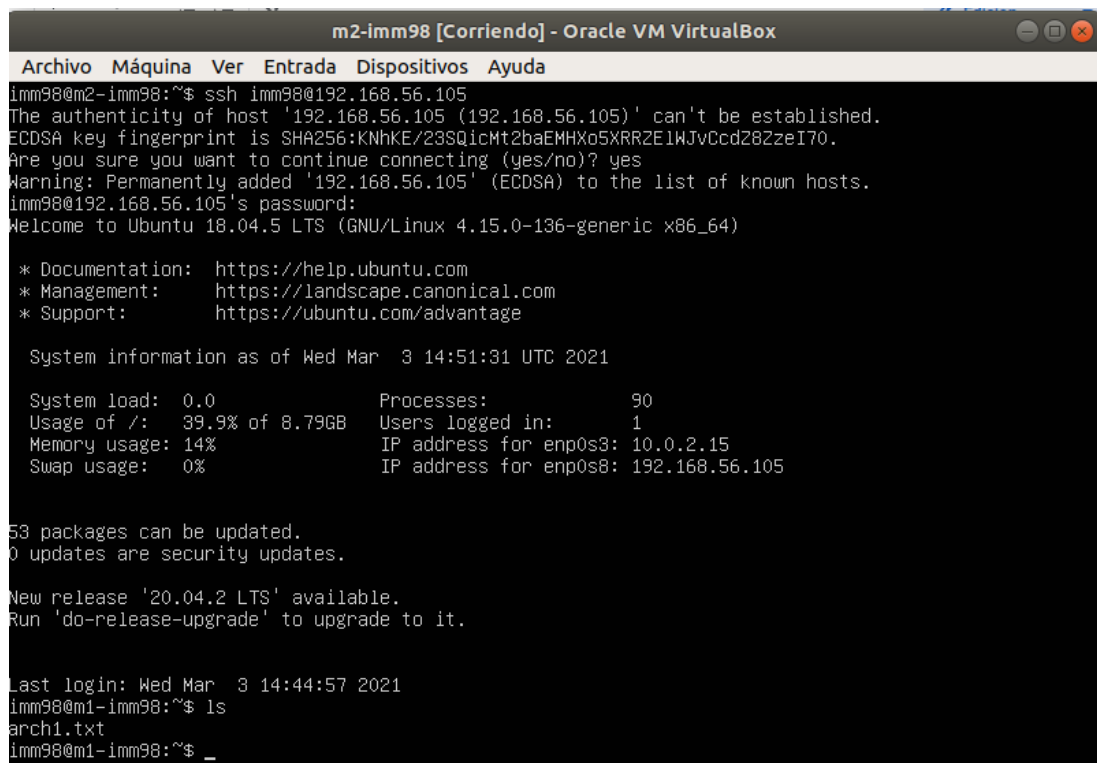
System load:  0.0               Processes:           94
Usage of /:   43.5% of 8.79GB   Users logged in:    1
Memory usage: 31%              IP address for enp0s3: 10.0.2.15
Swap usage:   0%               IP address for enp0s8: 192.168.56.104

53 packages can be updated.
0 updates are security updates.

New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Wed Mar  3 14:47:28 2021
imm98@m2-imm98:~$ ls
arch2.txt
imm98@m2-imm98:~$
```

- ssh de m2 a m1:



```
m2-imm98 [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
imm98@m2-imm98:~$ ssh imm98@192.168.56.105
The authenticity of host '192.168.56.105 (192.168.56.105)' can't be established.
ECDSA key fingerprint is SHA256:KNhKE/23SQicMt2baEMHXo5XRRZE1WJvCcdZ82zeI70.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.56.105' (ECDSA) to the list of known hosts.
imm98@192.168.56.105's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-136-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Wed Mar  3 14:51:31 UTC 2021

System load:  0.0               Processes:           90
Usage of /:   39.9% of 8.79GB   Users logged in:    1
Memory usage: 14%              IP address for enp0s3: 10.0.2.15
Swap usage:   0%               IP address for enp0s8: 192.168.56.105

53 packages can be updated.
0 updates are security updates.

New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Wed Mar  3 14:44:57 2021
imm98@m1-imm98:~$ ls
arch1.txt
imm98@m1-imm98:~$ _
```

El objetivo en ésta será ver cómo ejecutar comandos en equipos remotos mediante ssh sin contraseña. Para esta labor primero tendremos que crear una clave privada y una clave pública en cada máquina. Para ello utilizaremos el comando **ssh-keygen**:

**ssh-keygen -t rsa -b 4096**

- **-t**: Le indicamos que el tipo de clave que generamos va a ser de tipo **rsa**, por lo tanto se nos generará el fichero `~/.ssh/id_rsa` para la clave privada y fichero `~/.ssh/id_rsa.pub` para la clave pública
- **-b**: Indicamos el número de bits en la clave privada, en este caso 4096

```
imm98@m1-imm98:~/.ssh$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/imm98/.ssh/id_rsa):
/home/imm98/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/imm98/.ssh/id_rsa.
Your public key has been saved in /home/imm98/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:W44MBmyXdkTnkwWg6mrg8D1+5gk+wPnnk0Lo6B8BYGg imm98@m1-imm98
The key's randomart image is:
+---[RSA 4096]-----+
|O.      .+.O..      |
|+E .    + O O       |
|..  + = .  +        |
|   .. = .   .       |
|. O.. O S .         |
|++ O.. O =          |
|=++O. .  + .        |
|. +*=+=            |
|. +=B*..           |
+-----[SHA256]-----+
```

Una forma sencilla de hacer la copia de la clave a la otra máquina virtual es mediante el comando **ssh-copy-id** que copia la clave pública en este caso ubicada en `~/.ssh/id_rsa.pub` al archivo `~/.ssh/authorized_keys` de la otra máquina virtual, en este caso de `m2-imm98`. Por tanto ejecuto la orden:

**ssh-copy-id imm98@192.168.56.104**

```
imm98@m1-imm98:~/.ssh$ ssh-copy-id imm98@192.168.56.104
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/imm98/.ssh/id_rsa.pub"
The authenticity of host '192.168.56.104 (192.168.56.104)' can't be established.
ECDSA key fingerprint is SHA256:00QSWie5N0dv5zZ7zqzEEdw1J8kvzCCTPPQ2NCh0JLA.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are alr
eady installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to inst
all the new keys
imm98@192.168.56.104's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'imm98@192.168.56.104'"
and check to make sure that only the key(s) you wanted were added.
```

Comprobamos que se ha copiado dicha clave pública al archivo ~/.ssh/authorized\_keys de la máquina m2-imm98:

```
imm98@m2-imm98:~/ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQcDNTvvjDsSaHIbmw5Kt8TVc5PyvzdZqjja4wCUx8GubFXu/TJwJEuDV01V71cu
ZQrL1Ut/2uq2jpw/MdHmBmY+3+UAAuoTdYhSYdsVmk4cs/BwUYJXpdyAgxemRmTLQJ6EDIm5hnM0WaKM1te7wFtw7id/6R5E5E1d
fV3gCIE3vRCXfT3XqMf8bkVzbeSM1F1zrwPCnp8I/kDRYwg7Mp0D1XnVA910/FDaZEGRCBu9JSNf+pcLFQ842CE9PWSnf0B1nm15
ta9885d1VUp3W4x2WuKw+rz1xf3qGo/Laea801+qLHZgTksr9BEYBvQgJ/i10cX01wGz+s3Wt22DvFkruFw7FM4VkoagNWDEI3W0
X+qqfmeYVbIXCHpuFIjxtAdCkthDmKueYzSMDS2v3Z9ku+wJWhmmS3NtYIT25pwr2tQ39jNAj1fcRIWeQuPxywj6HptMjot1Cg9
iDolF8SHK/VCTH6BPevKUSF4n0PRYaSiVbU3002uAQ32vCa+3ntX+kt4Cdgd2Dg/q90z53G/kY5pz516fnyI+iidFE5zUYKYbd5v5
m1et9J45bLLMBra1Z6G5Apzim4bH6ZAwki+H00mjD6740+5WxS/8to/2ga29ZgKntDSbaIztHReD61no4Y1nAKZ3pIwmqXaSYGYU
QRXAmAsiWSfob/I1TS1ejw== imm98@m1-imm98
```

Por tanto probamos a ver si podemos conectarnos mediante ssh desde la máquina m1-imm98 a m2-imm98 y que no nos pide contraseña:

```
imm98@m1-imm98:~/ssh$ ssh imm98@192.168.56.104
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-136-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Tue Mar 16 17:55:40 UTC 2021

System load:  0.0               Processes:           94
Usage of /:   44.3% of 8.79GB   Users logged in:    1
Memory usage: 31%              IP address for enp0s3: 10.0.2.15
Swap usage:   0%               IP address for enp0s8: 192.168.56.104

 * Introducing self-healing high availability clusters in MicroK8s.
   Simple, hardened, Kubernetes for production, from RaspberryPi to DC.

   https://microk8s.io/high-availability

64 packages can be updated.
10 updates are security updates.

New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Mar 16 17:54:11 2021 from 192.168.56.105
imm98@m2-imm98:~$
```

Para realizar este mismo proceso pero sin el uso de la herramienta ssh-copy-id lo haremos con el siguiente comando, que redirige el contenido del archivo ~/.ssh/id\_rsa.pub de m1-imm98 al archivo ~/.ssh/authorized\_keys de m2-imm98:

`cat ~/.ssh/id_rsa.pub | ssh imm98@192.168.56.104 "cat >> ~/.ssh/authorized_keys"`

```
imm98@m1-imm98:~/ssh$ cat ~/.ssh/id_rsa.pub | ssh imm98@192.168.56.104 "cat >> ~/.ssh/authorized_keys"
imm98@192.168.56.104's password:
imm98@m1-imm98:~/ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDMFIJEXKgu+h2RWIebCVv1pfFTBGvFu44q4P/7qDYgJS2mSTaoVFFIgnqs6V9
LG4/zoAbYq5LcTae3PuF6mdTG2LGzESBbBC5e3d8AS+09opE0kfC4hfVL7+x4rDNNCNC8slzAZGwDiFuH1iCkf4HczrYo10+rpdtL
IYeE8I71Qf/71WMUgPQQmPJv4UeD7/Y2L20dsdv7ottS+2nke7pZiSjeFVvKQPMiKAgoTD+zh0xcHTMsitQ2VQEX3iv0IoIxpUG
m+1Ja9r782IaB2DUrukx8bkEBPgjrToGgYC3gjIhmyT/WaUL5E7PsJ8F11Xhf5SIg3/T8Gm508wux6wnF imm98@m1-imm98
imm98@m1-imm98:~/ssh$ _

m2-imm98 [Corriendo] - Oracle VM VirtualBox
archivo Máquina Ver Entrada Dispositivos Ayuda
imm98@m2-imm98:~/ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDMFIJEXKgu+h2RWIebCVv1pfFTBGvFu44q4P/7qDYgJS2mSTaoVFFIgnqs6V9
LG4/zoAbYq5LcTae3PuF6mdTG2LGzESBbBC5e3d8AS+09opE0kfC4hfVL7+x4rDNNCNC8slzAZGwDiFuH1iCkf4HczrYo10+rpdtL
IYeE8I71Qf/71WMUgPQQmPJv4UeD7/Y2L20dsdv7ottS+2nke7pZiSjeFVvKQPMiKAgoTD+zh0xcHTMsitQ2VQEX3iv0IoIxpUG
m+1Ja9r782IaB2DUrukx8bkEBPgjrToGgYC3gjIhmyT/WaUL5E7PsJ8F11Xhf5SIg3/T8Gm508wux6wnF imm98@m1-imm98
imm98@m2-imm98:~/ssh$ _
```

Una opción que permite el comando `ssh-keygen` es que las clave privada y la clave pública que genera se almacenen en un archivo cuyo nombre le indicamos como parámetro. Esto se hace con la orden **`ssh-keygen -f`**. Vamos a crear unas claves para que se almacenen en ficheros llamados `mykey` y `mykey.pub`:

```
ssh-keygen -f mykey
```

Y una vez generadas esas claves con la opción **`ssh-copy-id -i`** indicándole el nombre del archivo donde se encuentra la clave privada copiamos dicha clave pública en el archivo `~/.ssh/authorized_keys` de la otra máquina virtual:

```
ssh-copy-id -i ~/.ssh/mykey imm98@192.168.56.104
```

Vemos que se ejecuta correctamente:

```
imm98@m1-imm98:~/.ssh$ ssh-keygen -f mykey
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in mykey.
Your public key has been saved in mykey.pub.
The key fingerprint is:
SHA256:2TVNZUX3+NEMfn2AJfE/2PWmkdHo6aB9Ww8zoUTJ2L0 imm98@m1-imm98
The key's randomart image is:
+---[RSA 2048]---+
|
|      o+=oB|
|      +.O O=|
|      . B O.X|
|      o o ooO=|
|      S . o.Eo=|
|      + + =.|
|      . o B .|
|      . *.|
|      . .|
+-----[SHA256]-----+
imm98@m1-imm98:~/.ssh$ ls
authorized_keys  id_rsa  id_rsa.pub  known_hosts  mykey  mykey.pub
imm98@m1-imm98:~/.ssh$ ssh-copy-id -i ~/.ssh/mykey imm98@192.168.56.104
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/imm98/.ssh/mykey.p
ub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any
that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now
it is to install the new keys
key_load_public: invalid format
Load key "/home/imm98/.ssh/id_rsa": invalid format
imm98@192.168.56.104's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'imm98@192.168.56.104'"
and check to make sure that only the key(s) you wanted were added.
```

El contenido del archivo `~/.ssh/mykey.pub` en `m1-imm98` es el siguiente:

```
imm98@m1-imm98:~/.ssh$ cat mykey.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQACzwjtYBHWG1TTzwec08JWDT/lotZag7CiqNe1zKWT0GHfG5WA
yCHkX3RQhLhpmjgBRvvl3JWm06Rou9EuuOFJMWpzAwo6dAUieEVOqnGPcPJac9ZLOT1+6idrAWdLtIofS1yrubP
/YNfzNsQnoJFFftys1nRVQZKF8rARbiEpFdBEfBUiytP5Tb+d4L7sgtqNd6p7RMgDn+EazXuXjJhUVu3MJPEZ0V
r4COR6QvKT0C2tz4W/6bl0dBNa1715B7uHQeJgKMdzVEi/qrg2QWrw+YBVzT6dVo7taNyZc/4o19bSHF5yPGs5K
MnkZp5Eq12R5TYC0N0aMEXJI4DZhdWzn imm98@m1-imm98
```

y vemos que coincide con el contenido del archivo `~/.ssh/authorized_keys` en m2-imm98:

```
imm98@m2-imm98:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQACzwjYBHWG1TTzwec08JWDT/1ot2ag7CiqNe1zkWTOGHfG5WAYCHKX3RQhLhpm
JgBRvv13JWm06Rou9EuufJMWpzAwo6dAUieEV0qnGPcPJac9ZL0T1+6idrAWdLtIofS1yruBP/YNfzNsQnoJFFtys1nRVQZKF8
rArbiEpFdBEfBUiytP5Tb+d4L7sgtqNd6p7RMgDn+EazXuXjJhUVu3MJPEZ0Vn4COR6QvKT0C2tz4W/6b10dBNal715B7uHQeJgK
MdZVEi/qrg2QWrw+YBVzT6dVo7taNyZc/4o19bSHF5yPGs5KMnk2p5Eq12R5TYC0N0aMEXJI4D2hDWzn imm98@m1-imm98
```

Ahora para conectarnos sin contraseña desde m1-imm98 a m2-imm98 debemos de utilizar **ssh -i** indicándole el archivo donde se encuentra la clave. En este caso sería:

`ssh -i ~/.ssh/mykey imm98@192.168.56.104`

```
imm98@m1-imm98:~/.ssh$ ssh -i ~/.ssh/mykey imm98@192.168.56.104
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-136-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Tue Mar 16 18:28:31 UTC 2021

System load:  0.0               Processes:           94
Usage of /:   44.3% of 8.79GB   Users logged in:    1
Memory usage: 31%              IP address for enp0s3: 10.0.2.15
Swap usage:   0%               IP address for enp0s8: 192.168.56.104

 * Introducing self-healing high availability clusters in MicroK8s.
   Simple, hardened, Kubernetes for production, from RaspberryPi to DC.

   https://microk8s.io/high-availability

64 packages can be updated.
10 updates are security updates.

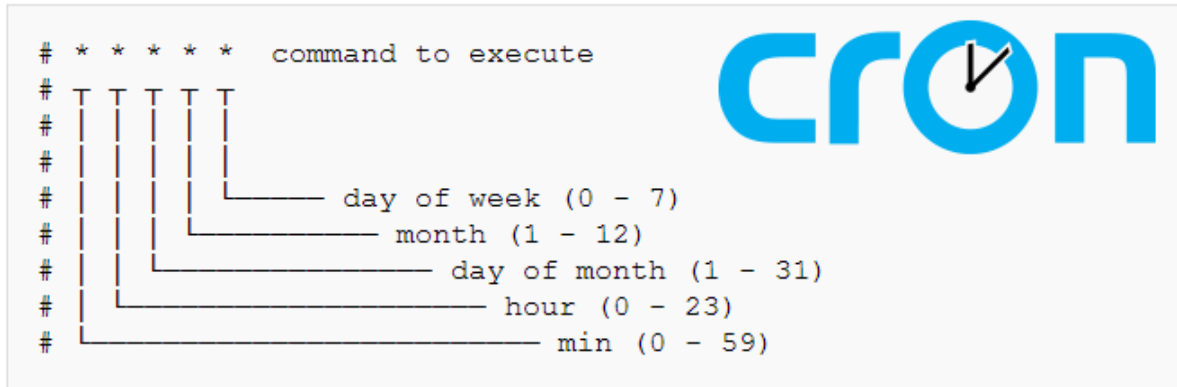
New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Mar 16 17:55:40 2021 from 192.168.56.105
```

# PROGRAMAR TAREAS CON CRONTAB

**cron** es un administrador de procesos en segundo plano que revisa cada minuto la tabla del fichero `/etc/crontab` en busca de tareas que se deban ejecutar.

Las tareas tendrán un formato de 7 campos:



estos 6 y el usuario que va a ejecutar las tareas.

Hay dos formas de añadir tareas para que las ejecute el demonio cron:

- Modificar el archivo `/etc/crontab` y añadir las tareas que queremos que se ejecuten y su frecuencia:
- Ejecutar el comando `crontab -e` y añadir ahí las tareas

Voy a probar los dos procedimientos. Voy a añadir al archivo `/etc/crontab` la siguiente línea para que cada 3 minutos borre todos los archivos del directorio `/tmp`

```
*/* * * * * root rm /tmp/*
```

```
imm98@m1-imm98:/etc$ cat crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cr
on.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cr
on.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cr
on.monthly )
30 0 * * * root    /sbin/shutdown -h now
30 15 * * 4 root    rm /tmp/*
*/3 * * * * root    rm /tmp/*
#
```



Voy a comprobar que efectivamente el demonio cron realiza esta tarea cada 3 minutos:

```
imm98@m1-imm98:/tmp$ touch hola.txt
imm98@m1-imm98:/tmp$ ls
hola.txt
systemd-private-97022cfede574479b8e77cf081790957-apache2.service-UUzmjQ
systemd-private-97022cfede574479b8e77cf081790957-systemd-resolved.service-4aE05I
systemd-private-97022cfede574479b8e77cf081790957-systemd-timesyncd.service-gYzRjM
imm98@m1-imm98:/tmp$ ls
systemd-private-97022cfede574479b8e77cf081790957-apache2.service-UUzmjQ
systemd-private-97022cfede574479b8e77cf081790957-systemd-resolved.service-4aE05I
systemd-private-97022cfede574479b8e77cf081790957-systemd-timesyncd.service-gYzRjM
```

Otro ejemplo sería que introdujera la fecha actual en un archivo llamado fecha.txt en el directorio home la fecha actual cada minuto:

```
imm98@m1-imm98:/etc$ cat crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
30 0 * * * root    /sbin/shutdown -h now
*/3 * * * * root    rm /tmp/*
* * * * * root    date >> /home/imm98/fecha.txt
#
```

y vemos que se realiza correctamente:

```
imm98@m1-imm98:~$ cat fecha.txt
Wed Mar 17 20:55:01 UTC 2021
Wed Mar 17 20:56:01 UTC 2021
Wed Mar 17 20:57:01 UTC 2021
Wed Mar 17 20:58:01 UTC 2021
Wed Mar 17 20:59:01 UTC 2021
```

Como hemos comentado con anterioridad, la otra opción para añadir tareas para que las ejecute el demonio cron es con **crontab -e**. Se nos abrirá un archivo donde escribiremos los comandos o scripts que queremos que utilice cron con el formato descrito anteriormente:

```
imm98@m1-imm98:~$ crontab -e
```

Primeramente vamos a ver que se ejecutan correctamente las tareas que introducimos en este archivo. Para ello introducimos una tarea para que almacene la fecha actual en un archivo llamado fichd.txt en el directorio home la fecha actual cada minuto. Para ver las tareas que hemos introducido ejecutamos **crontab -l**

```
imm98@m1-imm98:~$ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
* * * * * date > /home/imm98/fichd.txt
```

y comprobamos que esta tarea se ha ejecutado correctamente:

```
imm98@m1-imm98:~$ cat fichd.txt
Thu Mar 18 09:16:01 UTC 2021
```

Para borrar todas las tareas que hemos introducido mediante el comando **crontab -e** ejecutaremos la orden **crontab -r**

```
imm98@m1-imm98:~$ crontab -r
imm98@m1-imm98:~$ crontab -l
no crontab for imm98
imm98@m1-imm98:~$
```

Una vez que hemos comprobado el correcto funcionamiento de cron vamos a realizar lo que nos pide la práctica: Establecer una tarea en cron que se ejecute cada hora para mantener actualizado el contenido del directorio /var/www entre las dos máquinas.

Primeramente vamos a programar una tarea cada minuto (simplemente es para ver el correcto funcionamiento del comando con cron) para que copie el contenido de la carpeta /var/www/ de m2-imm98 en /var/www/ de m1-imm98.



Estos eran los contenidos de esa carpeta previos a la introducción de dicha tarea:

m1-imm98:

```
imm98@m1-imm98:/var/www/html$ ls
index.html
```

m2-imm98:

```
imm98@m2-imm98:/var/www/html$ ls
ejemplo.html  ejemplovacio.html  index.html  index2.html
```

Introduzco la siguiente tarea con crontab -e:

```
* * * * * rsync -avz -e ssh 192.168.56.104:/var/www/ /var/www/
```

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
* * * * * rsync -avz -e ssh 192.168.56.104:/var/www/ /var/www/
```

y compruebo que se ejecuta correctamente ya que se actualiza el contenido de /var/www/ en m1-imm98:

```
imm98@m1-imm98:/var/www/html$ crontab -e
no crontab for imm98 - using an empty one
crontab: installing new crontab
imm98@m1-imm98:/var/www/html$ ls
index.html
imm98@m1-imm98:/var/www/html$ ls
ejemplo.html  ejemplovacio.html  index.html  index2.html
```

Con la opción **crontab -u** especificamos el usuario cuyo archivo crontab se va a ver o modificar. Por tanto con el comando:

```
crontab -l -u imm98
```

se mostrará el archivo crontab del usuario con el que estoy realizando estas prácticas

```
imm98@m1-imm98:/etc/cron.d$ sudo crontab -l -u pepe
crontab: user `pepe' unknown
imm98@m1-imm98:/etc/cron.d$ sudo crontab -l -u imm98
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
* * * * * rsync -avz -e ssh 192.168.56.104:/var/www/ /var/www/
```

En segundo lugar vamos a programar una tarea cada minuto para que copie el contenido de la carpeta /var/www/ de m1-imm98 en /var/www/ de m2-imm98, al revés de como lo habíamos hecho previamente.

En el directorio /var/www/html de m1-imm98 tenemos esto:

```
imm98@m1-imm98:/var/www/html$ ls
ejemplo.html  ejemplovacio.html  index.html  index2.html  prueba.html
```

Introducimos mediante la orden crontab -e la siguiente tarea en m2-imm98

```
* * * * * rsync -avz -e 'ssh -p 2222' 192.168.56.105:/var/www/ /var/www/
```

```
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
* * * * * rsync -avz -e 'ssh -p 2222' 192.168.56.105:/var/www/ /var/www/
imm98@m2-imm98:/var/www/html$ ls
ejemplo.html  ejemplovacio.html  index.html  index2.html
imm98@m2-imm98:/var/www/html$ ls
ejemplo.html  ejemplovacio.html  index.html  index2.html  prueba.html
```

y vemos que efectivamente realiza esa tarea correctamente.

Una vez que hemos visto que cron ejecuta esa tarea correctamente modificamos las tareas creadas previamente para que en vez de ejecutarlas cada minuto las ejecute cada hora (es decir, el primer minuto de cada hora). Por lo tanto habría que escribir en m1-imm98:

```
0 * * * * rsync -avz -e ssh 192.168.56.104:/var/www/ /var/www/
```

```
0 * * * * rsync -avz -e ssh 192.168.56.104:/var/www/ /var/www/
```

y en m2-imm98:

```
0 * * * * rsync -avz -e 'ssh -p 2222' 192.168.56.105:/var/www/ /var/www/
```

```
0 * * * * rsync -avz -e 'ssh -p 2222' 192.168.56.105:/var/www/ /var/www/
```