COPIAS DE SEGURIDAD EN SISTEMAS LINUX

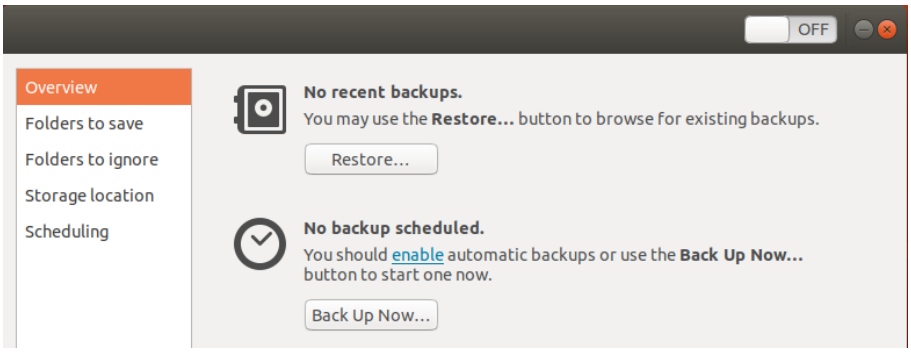
Índice

1. INTRODUCCIÓN

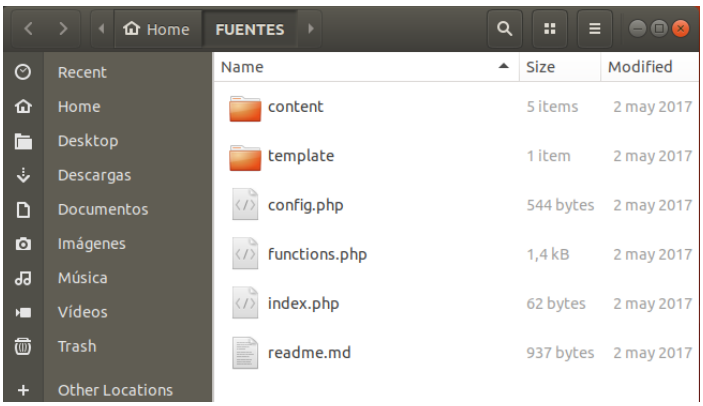
El tratamiento de las copias de seguridad está integrado en el propio SO en Linux, ya que con algunos de los comandos que incorpora podemos configurar perfectamente nuestros backups. Todas estas utilidades las incluyen la gran mayoría de distribuciones Linux por defecto, luego no es necesario instalar ningún Software específico, o en todo caso se encuentran en los repositiorios oficiales de las distintas distribuciones. Para realizar la CS podemos elegir entre multitud de opciones de comandos, y dentro de cada comando, opciones de ejecución. En esta práctica utilizaremos las siguientes herramientas:

A) TAR
B) RSYNC
C) CRON

La versión de Linux que hemos utilizado para mostrar los ejemplos es Ubuntu Server 18.04, pero al tratarse de comandos muy comunes, podrías elegir la distribución que quieras para realizar tu práctica. Aunque ubuntu 18.04 incluye un sencillo SW de copia de seguridad con GUI, es más interesante realizarlo con comandos y aprovechar las posibilidades que nos ofrecen.



Usamos el ejemplo que tenemos en el enunciado de la práctica(simple-php-website-master.zip), el cual es un sitio web básico programado en PHP que henmos descargado de GitHub. Lo descargamos y descomprimimos en una carpeta llamdas FUENTES. Finalmente debe quedar algo así en vuestro directorio home.



2. TAR

Por ahorrar espació, vamos a realizar una copia comprimida de nuestra carpeta FUENTES, además el utilizar este programa nos permitiría, aunque no vayamos a verlo en esta práctica, añadir al fichero un codigo de comprobación MD5 para verificar sus datos. Los ficheros comprimidos los guardaremos en una carpeta BACKUP.

A) Creación de una copia completa

Para crear una copia de seguridad completa, utilizaremos una orden como:

```
$tar -cvzf BACKUP/cseg-fuentes_TOTAL-`date +%Y-%m-%d`.tar.gz FUENTES
```

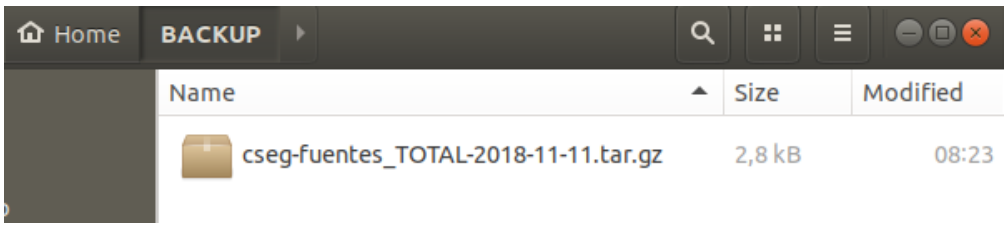
Los detalles de las opciones son:

- c - creando archivo (la opción x sería par extraer)
- v - modo detallado (verbose)
- f - fichero a generar
- z - tipo de compresión(gzip)
- `date +%Y-%m-%d` - añade al nombre del fichero la fecha de realización de la copia. Atención a las comillas (`)

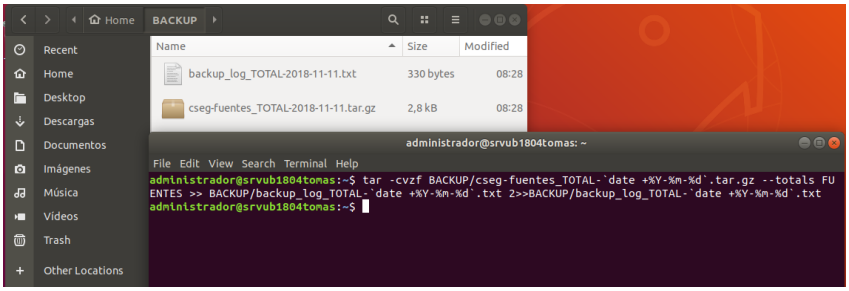
La ejecución del comando nos devolverá la información de los ficheros y carpetasque ha añadido al fichero comprimido.

```
administrador@srvub1804tomas:~$ tar -cvzf BACKUP/cseg-fuentes-`date +%Y-%m-%d`.tar.gz --totals FUENTES
FUENTES/
FUENTES/index.php
FUENTES/config.php
FUENTES/functions.php
FUENTES/template/
FUENTES/template/template.php
FUENTES/content/
FUENTES/content/products.php
FUENTES/content/404.php
FUENTES/content/about-us.php
FUENTES/content/home.php
FUENTES/content/contact.php
FUENTES/readme.md
Total bytes written: 20480 (20KiB, 9,1MiB/s)
```

Con esto ya tendemos creado nuestra copia de seguridad TOTAL, como vemos en la siguiente imagen. El fichero tiene incluida la fecha de ejecución de la copia:



pero puede ser muy interesante, y necesario en caso de problemas, el guardar también el resultado de la ejecución en un fichero de LOG, para ellos utilizaremos las redirecciónes que nos proporciona el kernel de Linux. En este caso comprobad los ficheros que ha generado la copia.



Otra opción muy interesante para asegurarnos de la corrección de nuestra copia de seguridad sería añadir al fichero un codigo de comprobación MD5 para verificar sus datos. Esta opción la utilizaremos más adelante.

```
administrador@srvub1804tomas:~$ md5sum BACKUP/cseg-fuentes_TOTAL-2018-11-11.tar.gz
e585cc78650c1a45f37b50b515f3971e  BACKUP/cseg-fuentes_TOTAL-2018-11-11.tar.gz
```

B) Creación de copias diferenciales

Para crear una copia de seguridad diferencial podemos usar la opcion -N (newer) de tar, tomando como punto de referencia la fecha de creación de la copia de seguridad total. Por ejemplo, si queremos hacer una copia de seguridad de nuestra carpeta fuentes, pero únicamente de los ficheros que se hayan modificado después del 1 de enero del 2019, usaríamos las siguientes opciones.

```
$tar -cvzf BACKUP/cseg-fuentes_DIFERENCIAL-`date +%Y-%m-%d` .tar.gz FUENTES -N 01ene2019
```

La salida de un comando como el anterior nos devolvería algo como lo siguiente, vemos que en realidad no copia ningún fichero:

```
administrador@srvub1804tomas:~$ tar -cvzf BACKUP/cseg-fuentes_DIFERENCIAL-`date +%Y-%m-%d`.tar.gz --totals FUENTES -N 2019-11-11
tar: Option --after-date: Treating date '2019-11-11' as 2019-11-11 00:00:00
FUENTES/
tar: FUENTES/index.php: file is unchanged; not dumped
tar: FUENTES/config.php: file is unchanged; not dumped
tar: FUENTES/functions.php: file is unchanged; not dumped
FUENTES/template/
tar: FUENTES/template/template.php: file is unchanged; not dumped
FUENTES/content/
tar: FUENTES/content/products.php: file is unchanged; not dumped
tar: FUENTES/content/404.php: file is unchanged; not dumped
tar: FUENTES/content/about-us.php: file is unchanged; not dumped
tar: FUENTES/content/home.php: file is unchanged; not dumped
tar: FUENTES/content/contact.php: file is unchanged; not dumped
tar: FUENTES/readme.md: file is unchanged; not dumped
Total bytes written: 10240 (10KiB, 3,9MiB/s)
```

En nuestro ejemplo, si queremos que haga algo como "coge los archivos que se hayan modificado con posterioridad a la creación de la copia TOTAL", podríamos usar el comando STAT para obtener ese dato, con el comando:

```
administrador@srvub1804tomas:~$ stat -c %z BACKUP/cseg-fuentes_TOTAL-2018-11-11.tar.gz
2018-11-11 09:23:06.979634032 +0000
```

Poniendo todo junto en el mismo comando, quedaría algo como:

```
administrador@srvub1804tomas:~$ tar -cvzf BACKUP/cseg-fuentes_DIFERENCIAL-`date +%Y-%m-%d`.tar.gz --totals FUENTES -N "`stat -c
%z BACKUP/cseg-fuentes_TOTAL-*`"
tar: Option --after-date: Treating date '2018-11-11 09:23:06.979634032 +0000' as 2018-11-11 09:23:06.979634032
FUENTES/
tar: FUENTES/index.php: file is unchanged; not dumped
tar: FUENTES/config.php: file is unchanged; not dumped
tar: FUENTES/functions.php: file is unchanged; not dumped
FUENTES/template/
tar: FUENTES/template/template.php: file is unchanged; not dumped
FUENTES/content/
tar: FUENTES/content/products.php: file is unchanged; not dumped
tar: FUENTES/content/404.php: file is unchanged; not dumped
tar: FUENTES/content/about-us.php: file is unchanged; not dumped
tar: FUENTES/content/home.php: file is unchanged; not dumped
tar: FUENTES/content/contact.php: file is unchanged; not dumped
tar: FUENTES/readme.md: file is unchanged; not dumped
Total bytes written: 10240 (10KiB, 5,9MiB/s)
```

Si finalmente queremos redireccionar toda la salida de la ejecucion del comando a otro fichero de log, tendriamos:

```
administrador@srvub1804tomas:~$ tar -cvzf BACKUP/cseg-fuentes_DIFERENCIAL-`date +%Y-%m-%d`.tar.gz --totals FUENTES -N "`stat -c %z
 BACKUP/cseg-fuentes_TOTAL-*`" > BACKUP/backup_log_DIFERENCIAL-`date +%Y-%m-%d` 2>>BACKUP/backup_log_DIFERENCIAL-`date +%Y-%m-%d`
```

Ahora puedes comprobar los ficheros que se han generado entre los pasos A y B.

C) Creación de copias incrementales

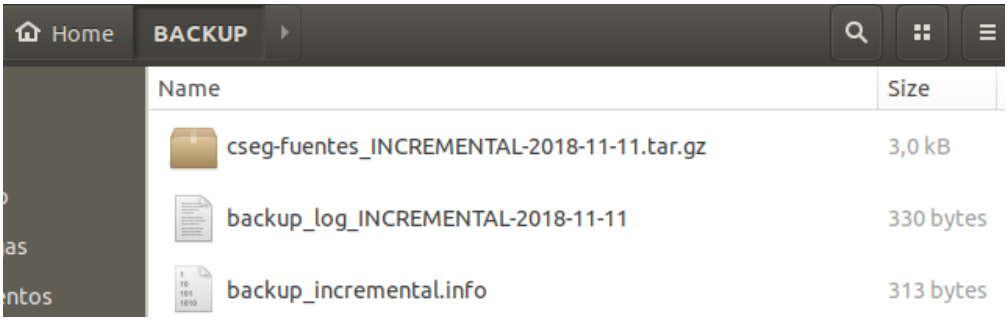
Por último, si lo que necesitamos en nuestro plan de copias de seguridad, es la creación de copias incrementales, podemos utilizar otra opción que nos proporciona TAR, en este caso el opción -g o –listed-incremental. Si recordamos lo visto en teoría las copias de seguridad incrementales se basan en la última copia de seguridad realizada y solo en las modificaciones realizadas desde la última copia de seguridad (completa, diferencial o incremental).

De acuerdo al manual de TAR, esta opción se utiliza de la siguiente manera:
"-g, --listed-incremental=ARCHIVO
ARCHIVO es el nombre de un archivo de instantáneas, donde tar almacena información adicional que se utiliza para decidir qué archivos han cambiado desde el anterior volcado incremental y, en consecuencia, se debe volcar nuevamente. Si ARCHIVO no existe, será creado y todos los archivos se agregarán al archivo resultante (El volcado de nivel 0), generando una copia de seguridad total. Para crear archivos incrementales de nivel N, crea una copia del archivo de instantánea creado durante el nivel N-1, y usarlo como ARCHIVO."

En nuestro caso, y siguiendo lo que indican en el siguiente enlace, podríamos ejecutar algo como lo siguiente, teniendo en cuenta que en la primera ejecución del comando TAR con la opcion incremental, el fichero de info no existe por lo que realizará una total:

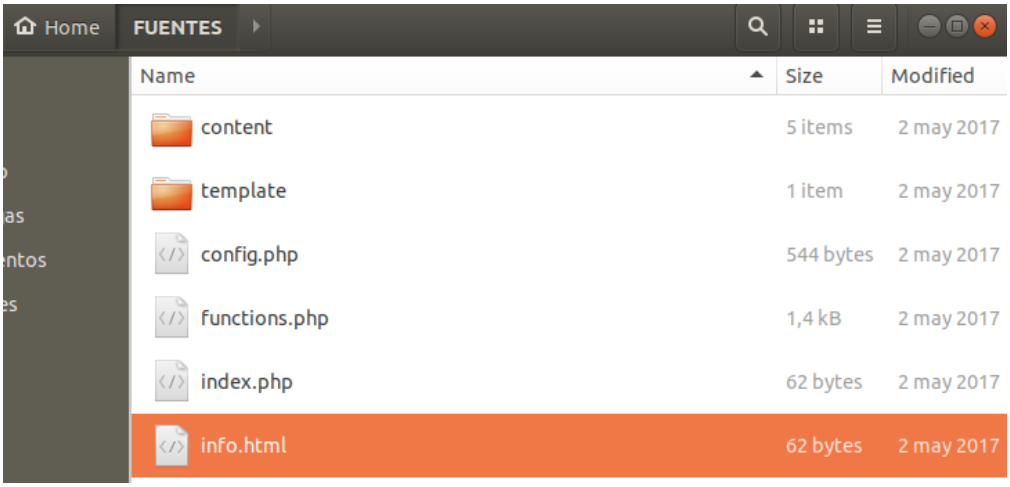
```
administrador@srvub1804tomas:~$ tar -cvzf BACKUP/cseg-fuentes_INCREMENTAL-`date +%Y-%m-%d`.tar.gz
-g BACKUP/backup_incremental.info --totals FUENTES > BACKUP/backup_log_INCREMENTAL-`date +%Y-%m-%d
` 2>>BACKUP/backup_log_INCREMENTAL-`date +%Y-%m-%d`
administrador@srvub1804tomas:~$
```

Lo que gener varios ficheros en la carpeta BACKUP con un contenido básico similar al de la copia total del apartado A, con la excepción del nuevo fichero .info:

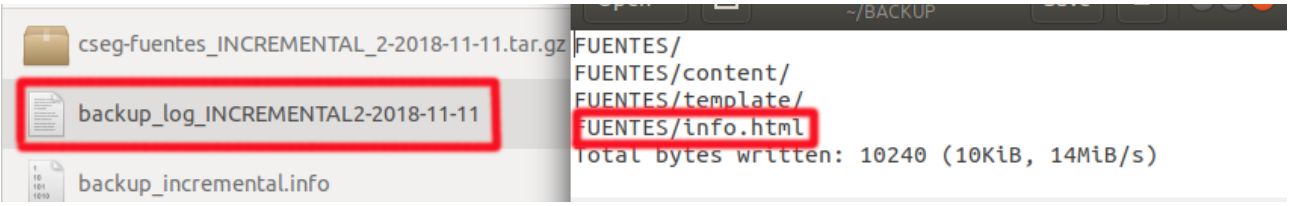


1. El fichero tar.gz: Copia de seguridad con TODO el contenido de la carpeta FUENTES
2. El fichero txt de log: Donde estoy almacenando el resultado de mis ejecuciones
3. El fichero .info: Contiene los datos de las carpetas y ficheros que se han copiado. Desde el generaremos las sucesivas copias incrementales. Ahora cada vez que generemos un fichero con tar y la opción -g tendremos en cuenta este .info.

En mi ejemplo, una vez creado la primera copia, voy a crear un niuevo fichero html que añadiré al directorio raiz de la carpeta FUENTES (*info.html*):



Al no existir información de ese fichero en el .info que antes he generado, si vuelvo a llamar al comando TAR con las opciones anteriores, pero con un NUEVO FICHERO DE SALIDA Y DE LOG PARA NO MACHACAR EL INICIAL, vemos que el único fichero que comprime en el nuevo tar.gz es el html que acabo de añadir, además de crear la estructura de directorios del sitio web del cual estoy creando la copia de seguridad.



Dependiendo del tipo de copias de seguridad que tengamos hechas, necesitaremos distintos ficheros , para realizar una restauración de los datos. Además, por comodidad, hemos usado el sistema de ficheros raiz (donde se encuentra la carpeta FUENTES original), para almacenar la carpeta BACKUP, en un entorno real deberíamos haber montado es BACKUP sobre un sistema de ficheros distinto del de sistema (otro disco duro, por ejemplo)

3. RSYNC

Se trata de una herramienta muy útil, con muchísimas opciones, que nos va a permitir cosas como:

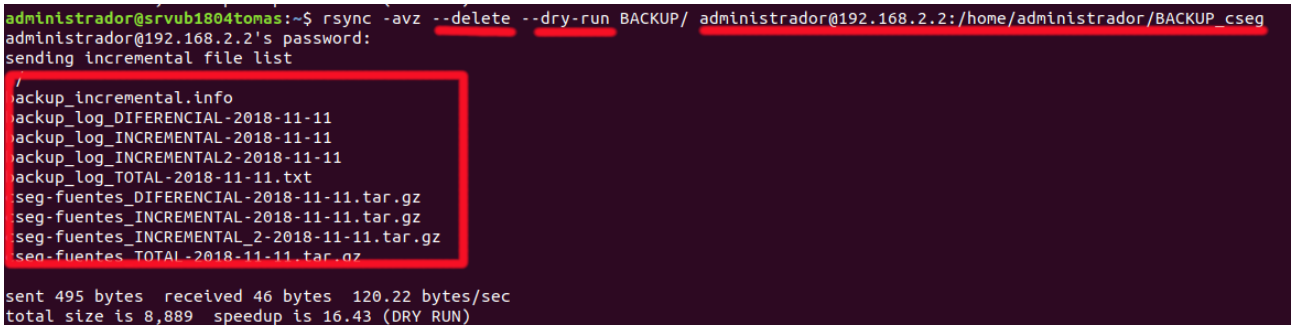
- Realizar copias de estructuras de directorios completas. Manteniendo sincronizadas ambas carpetas (la original y la copia)
- Elegir cómo se realiza esa copia(Total, diferencial, incremental)
- Mover esa copia a un equipo remoto usando un puerto seguro para cifrar el tráfico. Existen multitud de páginas donde podéis encontrar ejemplos y manuales con la explicación de las distintas opciones que podemos usar con este comando.

Vamos a mantener una copia de nuestra carpeta BACKUP sincronizada con un equipo en nuestra red local, para asegurarnos la redundancia que, en caso de fallo, nos permita seguir ejecutando nuestro servidor web ya que tendremos una copia de los datos.

Para poder realizar la simulación de la sincronización remota, necesitamos una segunda máquina de Ubuntu Server funcionando, y con conectividad con la MV que hemos utilizado en los pasos previos.

A) Sincronización de una carpeta local con un equipo remoto

Con un sencillo comando podemos mantener sincronizados los dos directorios, el BACKUP original y el que vamos a ubicar en una máquina remota. El comando a utilizar sería algo como (en la máquina remota he creado una carpeta BACKUP_cseg):



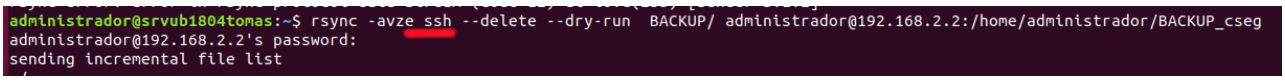
Podemos comentar varias de las opciones utilizadas:
-avz: Opciones por defecto, con ellas solo se copian las diferencias, muestra información de lo que se copia, se comprime la transferencia y se recorre recursivamente un directorio.
--delete: Elimina del directorio destino aquellos ficheros que ya no existan en el origen (por defecto los dejaría)
--dry-run: Ejecuta una simulación del comando, sin hacer realmente la transferencia. Útil para realizar pruebas.
Observamos que la ejecución del comando nos ha pedido la contraseña del usuario administrador en el equipo remoto.

B) Rsync sobre SSH para asegurar el cifrado de los datos transferidos

Para añadir más seguridad a este proceso, podemos realizar la transferencia con el comando ssh, siempre que en la máquina remota se encuentre en ejecución un servidor ssh. De esta manera la información viajaría cifrada, para ello usamos la opcion -e ssh. Para instalar el servidor ssh en el equipo remoto habría que ejecutar algo como:

```
$sudo apt-get install  openssh-server
```

El comando, añadiendo esta funcionalidad, quedaría en algo similar a la siguiente imagen:



C) Comprobación copia incremental

Para comprobar que únicamente se copian las diferencias, vamos a desactivar la opción –dry-run, con lo que el comando se ejecutara realmente.

```
administrador@srvub1804tomas:~$ rsync -avze ssh --delete BACKUP/ administrador@192.168.2.2:/home/administrador/BACKUP_cseg
administrador@192.168.2.2's password:
sending incremental file list
./
backup_incremental.info
backup_log_DIFERENCIAL-2018-11-11
backup_log_INCREMENTAL-2018-11-11
backup_log_INCREMENTAL2-2018-11-11
backup_log_TOTAL-2018-11-11.txt
cseg-fuentes_DIFERENCIAL-2018-11-11.tar.gz
cseg-fuentes_INCREMENTAL-2018-11-11.tar.gz
cseg-fuentes_INCREMENTAL_2-2018-11-11.tar.gz
cseg-fuentes_TOTAL-2018-11-11.tar.gz

sent 8,044 bytes  received 190 bytes  1,829.78 bytes/sec
total size is 8,889  speedup is 1.08
```

Una vez ejecutado si lo vuelvo a ejecutar inmediatamente, veo que no copia nada.

```
administrador@srvub1804tomas:~$ rsync -avze ssh --delete BACKUP/ administrador@192.168.2.2:/home/administrador/BACKUP_cseg
administrador@192.168.2.2's password:
Permission denied, please try again.
administrador@192.168.2.2's password:
sending incremental file list

sent 409 bytes  received 12 bytes  44.32 bytes/sec
total size is 8,502  speedup is 20.19
```

Si finalmente modifico un fichero de log y borro uno de los ficheros tar.gz, veo que solo se realizan esas dos modificaciones en la carpeta remota.

```
administrador@srvub1804tomas:~$ rsync -avze ssh --delete BACKUP/ administrador@192.168.2.2:/home/administrador/BACKUP_cseg
administrador@192.168.2.2's password:
sending incremental file list
deleting cseg-fuentes_INCREMENTAL_2-2018-11-11.tar.gz
./
backup_log_INCREMENTAL2-2018-11-11

sent 539 bytes  received 92 bytes  180.29 bytes/sec
total size is 8,502  speedup is 13.47
```

D) Evitar el poner el password en la ejecución de la sincronización.

Este paso puede sernos muy útil si queremos posteriormente automatizar lo máximo posible el proceso de copia de seguridad, lo más seguro sería utilizar un sistema de claves pública-privada, pero lo veremos más adelante por lo que vamos a utilizar un método más sencillo. Con el comando SSHPASS podemos pasarle esa contraseña directamente en la linea de comandos, sin tener que contestar ninguna pregunta. Para instalar el comando:

```
$sudo apt-get install  sshpass
```

Una vez instalado su uso es muy simple:

```
administrador@srvub1804tomas:~$ sshpass -p "Asix2018" rsync -avze ssh --delete BACKUP/ administrador@192.168.2.2:/home/administrador/BACKUP_cs
sending incremental file list

sent 409 bytes  received 12 bytes  280.67 bytes/sec
total size is 8,502  speedup is 20.19
```

Vemos que directamente se ha hecho la transferencia, en esta caso vacía.

4. CRON

En este punto tenemos nuestras copias de seguridad creadas y comprimidas, todas ellas incluidas en una carpeta local (BACKUP), la cual mantenemos sincronizcada con una carpeta remota en el segundo servidor Ubuntu.
La última herramienta que nos quedaría por conocer para poder elaborar nuestro plan de copias de seguridad sería el comando CRON.

En todos los sistemas Unix, cron es un administrador de procesos en segundo plano (demonio) que ejecuta procesos o scripts a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab. Cron se podría definir como el "equivalente" a Tareas Programadas de Windows. (fte:wikipedia). Podéis encontrar un manual bastante útil en el siguiente enlace.

A) Funcionamiento general

Básicamente vamos a querer hacer dos cosas con nuestrar tareas programadas, o listarlas o editarlas. Para ellos tenemos los siguientes comandos:

1) Listar las tareas programadas:

```
$sudo crontab -l
```

2) Para editar las tareas programadas:

```
$sudo crontab -e
```

Antes de nada debemos darle un vistazo al contenido del fichero CRONTAB:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01  * * * * root nice -n 19 run-parts /etc/cron.hourly
50  0 * * * root nice -n 19 run-parts /etc/cron.daily
22 4 * * 0 root nice -n 19 run-parts /etc/cron.weekly
42 4 1 * * root nice -n 19 run-parts /etc/cron.monthly
```

Una vez usamos la opción -e, ya podemos comenzar a crear tareas programadas, siguiendo lo que se explica en el punto siguiente.

B) Creación de tareas programadas

Las tareas que podemos programar con cron, pueden ser:

A) Un comando
B) Una serie de comandos (separados por ;)
C) Scripts

Todos esos asteriscos y números que hemos visto en el fichero CRONTAB..¿QUE SIGNIFICAN?, entenderlos es la clave para programar nuestras tareas (en nuestros casos los backup) correctamente.
Cada uno de esos asteriscos o numeros, separados por espacios o tabulaciones(preferiblemente espacios) representan magnitudes temporales distintas, las cuales las podríamos resumir de la siguiente manera:

```
.--------------- minuto (0-59)
|  .------------ hora (0-23)
|  |  .--------- día del mes (1-31)
|  |  |  .------ mes (1-12) o jan,feb,mar,apr,may,jun,jul... (meses en inglés)
|  |  |  |  .--- día de la semana (0-6) (domingo=0 ó 7) o sun,mon,tue,wed,thu,fri,sat (días en inglés)
|  |  |  |  |
*  *  *  *  *  comando a ejecutar
```

C) Ejemplos de uso

Unos pocos ejemplos pueden servir más que cualquier explicación para entender el funcionamiento. Por ejemplo:

```
45 5 * * 1 who >>usuariosconectados.txt
```

Estamos diciendo que queremos que el comando who muestre los usuarios conectados y dirija el resultado añadiendo a un fichero de texto. ¿Cuándo?

En el minuto 45
De las 5 de la noche
De cada día del mes
De cada mes
Sólo si es lunes

En resumen, todos los lunes a las 5:45 horas se ejecutará el comando.

Si queremos, también podemos establecer intervalos de tiempo:
Para ejecutar un script de lunes a sabado a las 5:45 horas:

```
45 5 * * 1-6 /bin/ejecutar/scriptdeturno.sh
```

En el caso de que necesitemos ejecutar un script de lunes a viernes cada 10 minutos desde las 3:00 horas durante una hora:

```
0,10,20,30,40,50 3 * * 1-5 /bin/ejecutar/scriptdeturno.sh
```

La sintaxis de crontab permite lo siguiente. Imaginemos que queremos ejecutarlo cada 5 minutos:

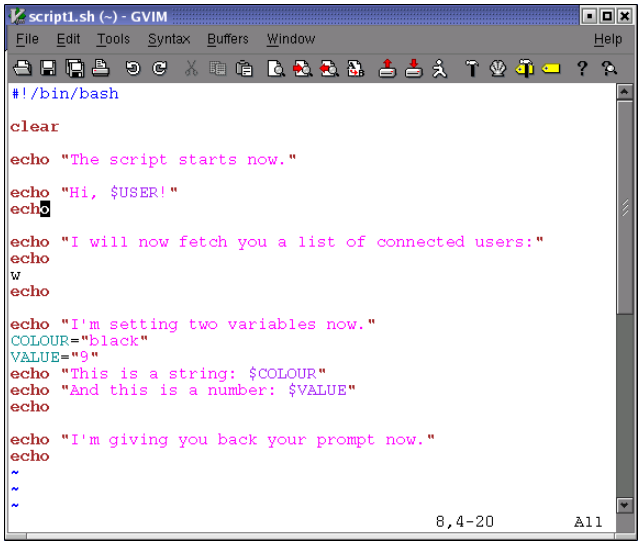
```
*/5 3 * * 1-5 /bin/ejecutar/scriptdeturno.sh
```

5. UN SCRIPT PARA GOBERNARLOS A TODOS

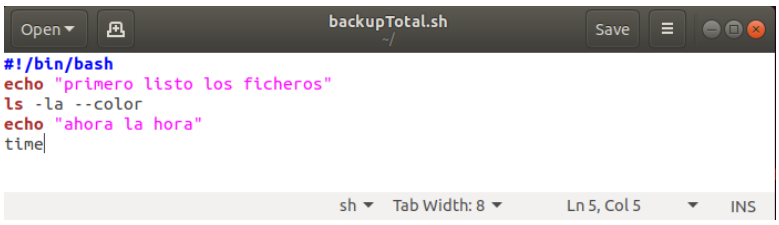
Conociendo los comandos que necesitamos, y como podemos programar su ejecución para el momento que queramos, lo recomendable sería escribir un script donde realizaramos nuestros backups, sincronizaramos con nuestra carpeta remota y además incluyamos comprobaciones de integridad (CHECKSUM) o borrado de copias antiguas automáticas. Finalmente programaríamos la ejecución de nuestro script con la utilidad CRON.
Pero antes de nada..¿qué es un script?
En este apartado nos referimos a un shell-script. Siendo una shell una ventana del intérprete de comandos, el cual es capaz de ejecutar las instrucciones linux que le proporcionemos, un script en este entorno es una agrupación ordenada de comandos, con los que se puede realizar comprobaciones y automatizar tareas complejas.
Para cualquier administrador de sistemas es fundamental tener nociones de shellscritping (trabaje con el SO que trabaje)

A) Estructura de un script en Linux

Un script es un fichero de texto que comienza con una linea en la que se índica el interprete de órdenes.



Es importante que el fichero tenga permisos de ejecución para que funcione.
Un sencillo ejemplo de prueba puede ser..



B) Un ejemplo para nuestro plan de copias de seguridad

Para nuestras copias de seguridad vamos a hacer lo siguiente:
A) Un script para las copias de seguridad total, el cual copiará todo el contenido de la carpeta fuente, además guardará el log de su actividad en un fichero de texto. Finalmente incluirá en

el fichero comprimido un checksum para que se pueda comprobar la integridad de la copia. Por último enviará todo a un servidor remoto.
B) Un scritp para las copias de seguridad diferenciales, que realizará los mismos pasos que el script anterior.
Después de codificarlos, programaremos la ejecución del script A los viernes a las 03:00 horas (por ejemplo) y la ejecución del script B el resto de días a la misma hora.

C) Código fuente

BACKUP TOTAL

```
#!/bin/bash
#capturo la fecha de hoy que luego voy a usar en los nombres de los ficheros
fecha=`date +%Y-%m-%d`
#variables de conexion..
user=administrador
password=Asix2018
IP_servidor=192.168.2.2
carpeta_remota=/home/administrador/BACKUP_cseg
tar -cvzf BACKUP/cseg-fuentes_TOTAL-$fecha.tar.gz --totals FUENTES >
BACKUP/backup_log_TOTAL-$fecha.txt 2>>BACKUP/backup_log_TOTAL-$fecha.txt
#añador al log el checksum..OBTENGO SOLO EL CÓDIGO MD5..NO EL NOMBRE DEL
FICHERO..lo dejo en un fichero llamado checsum
md5sum BACKUP/cseg-fuentes_TOTAL-$fecha.tar.gz | cut -d " " -f 1 >>
BACKUP/backup_log_TOTAL-$fecha.txt
#solo me falta sincronizar con la carpeta remota...
sshpass -p $password rsync -avze ssh --delete BACKUP/ $user@$IP_servidor:
$carpeta_remota >> /dev/null 2>>/dev/null
```

BACKUP DIFERENCIAL

```
#!/bin/bash
#capturo la fecha de hoy que luego voy a usar en los nombres de los ficheros
fecha=`date +%Y-%m-%d`
#capturo el instante de modificacion de la ULTIMA COPIA TOTAL para saber desde cuando
tengo que guardar
ultimacopiatotal=`ls -t BACKUP/cseg-fuentes_TOTAL* | head -1`
fechamodificacion=`stat -c %z $ultimacopiatotal`
#variables de conexion..
user=administrador
password=Asix2018
IP_servidor=192.168.2.2
carpeta_remota=/home/administrador/BACKUP_cseg
#EJECUTO MIS COMANDOS
tar -cvzf BACKUP/cseg-fuentes_DIFERENCIAL-$fecha.tar.gz --totals FUENTES -N
"$fechamodificacion" > BACKUP/backup_log_DIFERENCIAL-$fecha.txt
2>>BACKUP/backup_log_DIFERENCIAL-$fecha.txt
#añador al log el checksum..OBTENGO SOLO EL CÓDIGO MD5..NO EL NOMBRE DEL
FICHERO..lo dejo en un fichero llamado checsum
md5sum BACKUP/cseg-fuentes_DIFERENCIAL-$fecha.tar.gz | cut -d " " -f 1 >>
BACKUP/backup_log_DIFERENCIAL-$fecha.txt
#solo me falta sincronizar con la carpeta remota...
sshpass -p $password rsync -avze ssh --delete BACKUP/ $user@$IP_servidor:
$carpeta_remota >> /dev/null 2>>/dev/null
```

Solo quedaría realizar las configuraciones pertinentes en CRON