Copias de seguridad

Copias de seguridad en Linux

Generalmente en linux las copias se hacen utilizando una combinación de diferentes comandos que traen los sistemas operativos GNU/Linux. En este caso nos fijaremos en estos:

- `tar` para archivar documentos y carpetas
- `cron` para programar tareas
- `scp` para copias remotas

Con una combinación de dichos comandos, dentro de un script, podemos automatizar copias de seguridad.

Tar

A la hora de hacer copias nos resulta útil empaquetar varios archivos en uno solo, que será más manejable.

Tar toma una lista de archivos y/o carpetas y los guarda en un archivo concreto.

Para ello se puede utilizar el comando `tar`.

La sintaxis de `tar` es:

```
tar -argumentos archivo_a_crear archivos_a_copiar
```

Argumentos

Según que argumentos le pasemos al comando, podremos hacer diferentes cosas.

Crear un archivo:

- c: Crea un nuevo archivo que contiene los elementos especificados.
- f: Especifica el nombre del archivo.

Compresión:

- j: Se utiliza para comprimir el archivo a través de bzip2.
- z: Comprime un archivo a través de la herramienta gzip.
- x: Extrae el archivo de almacenamiento en el disco.

Otras opciones:

- v: Ejecuta el comando en modo detallado para mostrar el progreso del archivo.
- W: se utiliza para verificar un archivo comprimido.
- t: se usa para visualizar el contenido del archivo.
- r: usado para agregar o actualizar archivos o directorios al archivo existente.
- u: como -r, pero las nuevas entradas se agregan solo si tienen una fecha de modificación más nueva que la entrada correspondiente en el archivo.

Crear archivos de prueba

Nos vamos a nuestro home y creamos una carpeta de pruebas:

```
cd ~
mkdir docs
cd docs
```

Creamos archivos de prueba:

```
touch doc1 doc2 doc3
```

Estos son los archivos que utilizaremos para crear las copias.

Empaquetar archivos

Podemos empaquetar archivos (sin compresión) de la siguiente forma:

```
tar -cvf backup.tar *
tar -cvf backup.tar doc1 doc2 doc3
```

Ver los contenidos de un tar

Una vez que tengamos algún tar creado, podemos ver lo que hay dentro sin necesidad de extraer su contenido:

```
tar -tf backup.tar
```

Extraer contenido del tar

Vamos a borrar los documentos originales, simulando que los hemos perdido:

```
rm doc*
```

Probamos a extraer el contenido del tar.

```
tar -xvf backup.tar
```

Si todo va bien se nos habrán guardado.

Insertar y extraer archivos en un archivo tar (opcional)

Podemos meter y sacar archivos de un tar sin tener que extraerlo todo y volverlo a empaquetar.

Para borrar archivos de dentro de un tar existente:

```
tar -f backup.tar --delete file1 file2
```

Para meter archivos dentro del tar:

```
tar -rf backup.tar file1 file2
```

Comprimir una carpeta

También podemos comprimir el archivo `tar`. Se pueden utilizar dos mecanismos de compresión. Para comprimir un archivo `tar` con gunzip, se utiliza la extensión `.tar.gz`.

```
tar -cvzf backup-comprimido.tar.gz doc1 doc2 doc3
```

Ahora podemos comprobar la diferencia de tamaño entre el `.tar` (sin compresión) y el `.tar.gz`.

```
ls -lisa | grep tar
```

En caso que quisiéramos extraer el contenido:

```
tar -xvzf backup-comprimido.tar.gz
```

Utilizar fecha

En este caso estamos creando un `tar` comprimido que contendrá todos los archivos de la carpeta en la cual estamos.

```
tar -cvzf backup-$(date +%Y-%m-%d).tar.gz *
```

Cron

Otra utilidad importante es la de poder automatizar la ejecución de tareas sin que el usuario lo tenga que hacer manualmente. Para ello, se utiliza el daemon de `cron`.

`Cron` es un proceso que, consultando una lista de tareas, las va ejecutando cuando se cumplen ciertas condiciones de tiempo, como una hora concreta, un día concreto, etc.

La sintaxis para poder modificar esta información es:

```
crontab [-e] -l [-r] [usuario]
```

Parámetros de cron

- `e` indica la edicion del cron
- `l` ver las tareas programadas en el archivo cron
- `r` borrar un archivo cron

Edición del crontab

Cada usuario del sistema puede tener sus propias tareas para que cron las ejecute. Si no se especifica el usuario, el comando se ejecutara para el usuario que tiene iniciada la sesión.

```
crontab -e
```

Veremos que el contenido del archvo `crontab` está vacío, porque nunca lo hemos utilizado.

Por defecto, se nos abrirá para editar con `vim`, en modo de lectura.

- Para insertar cosas, tenemos que pulsar `i`.
- Para salir del modo edición, pulsamos `esc`.

- Para salir guardando, tras salir del modo edición, escribir `:wq` y pulsar intro.

Dentro del archivo introduciremos las lineas de texto con las diferentes tareas.

- Cada línea representa una tarea
- Cada tarea está formada por dos partes: el momento y la tarea a ejecutar.

El formato es el siguiente:

```
* * * * * comando_o_programa_a_ejecutar
```

Cada asterisco en orden significa:

- Minuto (0 -59)
- Hora (0 – 23)
- Día del mes (1 – 31)
- Mes (1 – 12)
- Día de la semana (0 – 6) 0 domingo

Ejemplos de cron

```
0 3 * * 5 /home/user/backup (ejecuta todos los viernes a las 3 de la mañana)
15 10 * * * /home/user/backup (ejecutar todos los días a las 10:15)
```

Envío remoto de la copia de seguridad

Lo más aconsejable es guardar las copias de seguridad en un dispositivo de almacenamiento o máquina diferente a la que contiene la información original. De este modo, si falla el disco original, no se destruirá la copia.

Para ello existen muchos comandos:

- ssh
- scp
- rsync

Cada uno de ellos se utiliza para cosas diferentes. Una de las posibilidades consiste en hacer la copia con `scp`.

```
scp /home/user/copiaseguridad.tar usuarioremoto@servidor:/etc/backups/
```

Combos

Normalmente se suelen combinar varios de los comandos vistos, junto con los comandos `ssh` o `scp` para, al mismo tiempo:

- Hacer un backup de la carpeta períodicamente
- Enviarlo a una localización remota

Reenviar salida de tar a stdout

Podemos evitar que `tar` cree un archivo de salida y, a cambio, redirija la salida a stdout.

```
tar cvzf - /path/to/myBase.sql | ssh USER@HOST "dd
of=/path/to/backups/myBase$(date +\%Y\%m\%d\%H\%M\%S).tar.gz"
```

Agregar comando a crontab

Con el siguiente comando, podemos agregar una tarea a `cron`.

```
crontab -e
04 * * * *
tar -zcvf "$(date '+%Y-%m-%d').tar.gz doc*
scp datadir-'date+\%s'.tar.gz user@serverremoto:/backups
```

Crear un script

Otra salida podría ser crear un `script` con todas las instrucciones y, simplemente, llamar al `script` dentro de cron.

Creamos un archivo `backup.sh`:

```
#!/bin/bash
tar -zcvf "$(date '+%Y-%m-%d').tar.gz doc*
scp datadir-'date+\%s'.tar.gz user@serverremoto:/backups
```

El siguiente comando devuelve la fecha utilizando subshell: `$(date '+%Y-%m-%d')`

Le damos permisos de ejecución:

```
chmod +x backup.sh
```

Modificamos el crontab:

```
crontab -e 30 11 * * * backup.sh
```

Copias incrementales

Podemos enviar copias incrementales utilizando rsync. Para ello:

```
rsync -avz ruta/a/carpeta userremoto@serverremoto:/backups
```