

Practica05

Partition to store the backups

1. Mirem que tinguem algun sector lliure en el disc per on poder crear la nova partició.

NOTA: Si no hi ha, podem fer resize ('e') a una mida més petita d'alguna existent

“**sudo fdisk <disk_name>**” > ‘F’

2. Creem la nova partició amb les mides que volem.

“**sudo fdisk <disk_name>**” > ‘n’ > ‘w’

```
Command (m for help): n
Partition number (1-128, default 1):
First sector (45090816-62914526, default 45090816):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (45090816-49285119, default 49285119):

Created a new partition 1 of type 'Linux filesystem' and of size 2 GiB.
```

3. Li assignem un filesystem a la nova partició

NOTA: Potser fa falta instal·lar “btrfs-progs”.

NOTA: Si surt aquest error “WARNING: failed to open /dev/btrfs-control, skipping device registration: No such file or directory” → Ignorar.

“**sudo mkfs -t btrfs <partition_name>**”

4. Creem la nova carpeta “/backup” amb els permisos corresponents.

“**mkdir /backup**” > “**chmod 700 /backup**” > “**sudo chown root:root /backup**”

5. Finalment muntem la partició a la carpeta.

“**sudo mount /dev/sda7 /backup**”

6. Si volem que tot això quedí guardat en disc, haurem d'actualitzar el “/etc/fstab”

d'aquesta forma:

NOTA: Executar “**sudo blkid**” per a veure el UUID.

NOTA: Treure “defaults” no implica perdre les opcions perquè aquestes ja eren les per defecte en el Kernel. En cas de col·lisió (ex: “rw” v.s. “ro”) guanya l'última escrita.

```
#UNCONFIGURED FSTAB FOR BASE SYSTEM
# <partition>          <directory to mount>    <file system>   <options>      <dump>  <checking order>
UUID="d4feba128-2f23-478e-bc02-7c4f35d90cc0"    none           swap        defaults     0       0
UUID="98e48cce-f1bc-4080-b981-af8d87e9d3dc"    /             ext4        defaults     0       1
UUID="9025-C8C8"          /boot/efi        vfat        defaults     0       2
UUID="9fe11e60-c178-4445-824f-0f9088fd76f0"    /usr/local      ext4        defaults     0       2
UUID="4102c54b-02a0-473b-aa20-01fd659f023b"    /home         ext4        defaults     0       2
UUID="a1fb60f6-92e4-4757-b102-f6a10d4e2f2e"     /backup        btrfs       ro        0       2
```

Making backups using “tar”

Full backup

1. Si volem crear una backup de tots els fitxers
 - “-c”: Compress.
 - “-f”: Filename.
 - “-p”: Preserve permissions.

“`sudo tar -cfp "/backup/backup_root_level0_$(date '+%d-%m-%Y').tar" /root`”

2. Si volguéssim comprimir-ho, hauríem d’afegir el flag “-z” per usar “gzip”.
Si ho fem, seria recomanable canviar a l’extensió “[tar.gz](#)”

Comprimir un backup, en termes de seguretat, no és una bona pràctica pel següent motiu: “Redueix l’entropia”.

Això significa que els algoritmes de compressió busquen patrons per reduir repeticions i, gràcies a l’estadística podríem arribar a trobar patrons comuns en els fitxers i obtindre informació que no voldrem que l’atacant tingui.

A sobre, podria ser que l’algoritme de compressió tingués alguna vulnerabilitat i es podria explotar.

3. En cas que volguéssim excloure alguns fitxers, podríem usar el paràmetre “[--exclude=PATTERN](#)” de “tar”.
IMPORTANT: Aquest ha d’anar a l’inici de la comanda.
4. Per temes de seguretat, és recomanable crear un fitxer “.asc” que contingui un SHA del backup per així després comprovar que no s’ha modificat.
NOTA: No fer servir “>” per crear el segon fitxer perquè en cas de “/backup” no tenim permis de redirecció.

“`sudo sha512sum <file> | sudo tee <file>.asc`”

Incremental backup

La data del fitxer resultant de fer la còpia (el .tar) indica l’hora la qual s’ha acabat de realitzar, NO l’hora d’inici del procés. Això pot fer que, si durant el procés algun usuari modifica un dels fitxers dins el directori que s’està fent la còpia → Hi hagi inconsistència a l’hora de restaurar l’incremental.

Per evitar-ho, podem crear un fitxer que marcarà (la seva pròpia creació) el timestamp abans d’iniciar el backup per així, quan restaurem, agafar l’hora de referència d’aquest i no el del fitxer “.tar”

1. Definim els noms dels fitxers (per comoditat)

“`export NAME_FULL=/backup/backup_root_level0_<data>`”

“`export NAME_INC=/backup/backup_root_level1_$(date '+%d-%m-%Y')`”

2. Creem el timestamp per aquesta nova còpia incremental.

NOTA: Suposarem que també s'ha fet en el cas de la total.

“sudo touch \$NAME_INC.timestamp”

3. Creem la nova còpia incremental respecte al timestamp que marca el fitxer de temps i no el propi de backup.

“sudo tar -cpf NAME_INC.tar --newer=NAME_FULL.timestamp /root”

4. Finalment també incloem el SHA512 resultant.

“sudo sha512sum NAME_INC.tar | sudo tee NAME_INC.asc”

Restoring a backup

IMPORTANT: “GNU tar” guarda el directori sense “/”, fent que en restaurar hagim d’anar a la ruta on hauria d’anar.

1. Anar a la ruta on volem que es restaurin les còpies i restaurar inicialment la total.

“cd /”

“sudo tar -xpf NAME_FULL.tar”

2. Quan ja tenim la total (la base) haurem de restaurar les incrementals EN ORDRE.

“sudo tar -xpf NAME_INC1.tar”

“sudo tar -xpf NAME_INC2.tar”

Notem que els fitxers eliminats no es perden perquè si han sigut eliminats entre la total i alguna incremental, és la total qui els conté.

En entorns professionals en les còpies s’inclou un “manifest” amb els noms de tots els fitxers. D’aquesta forma en restaurar podem verificar que la restauració ha sigut completa.

3. Si volguéssim restaurar només algun directori dins del backup, hauríem d’indicar-ho en la comanda tar.

Recorda que dins el tar estem parlant de rutes relatives.

“export FOLDER=root/fotos_personals”

“cd /”

“sudo tar -xpf NAME_GENERAL.tar FOLDER”

Making backups using “rsync”

Making backups over a network

1. Si volem fer servir “rsync” amb SSH primer haurem d’instal·lar aquest servei:
“**sudo apt update && sudo apt install -y openssh-server**”

2. Un cop instal·lat ens assegurarem que està funcionant.

NOTA: Si no ho està, canviar “status” per “start”.

“**systemctl status ssh.service**”

3. Un cop tenim tot preparat, haurem de permetre (per a la pràctica) que “root” estigui permès en les connexions ssh.

“**sudo vim /etc/ssh/sshd_config**”

“**sudo systemctl reload ssh.service**”

“**ssh root@localhost**”

```
# Authentication:  
  
#LoginGraceTime 2m  
PermitRootLogin yes  
#StrictModes yes
```

Making full backups

1. Primer creem una carpeta dins de “/backup” per a “rsync” i fem una primera copia total.

“-a”: Archive. És una macro que s’expandeix a més opcions (les especificades en el man)

“-v”: Verbose.

“-z”: Compress.

“**sudo mkdir /backup/rsync-backup**”

“**sudo rsync -avz /root -e ssh root@localhost:/backup/rsync-backup**”

```
aso@aso-client:~$ sudo mkdir /backup/rsync-backup  
aso@aso-client:~$ sudo rsync -avz /root -e ssh root@localhost:/backup/rsync-backup  
The authenticity of host 'localhost (::1)' can't be established.  
ED25519 key fingerprint is SHA256:pASYknSx7FkO28xGhI/1bNQaZjUrs0Fqn9qN8b9c5sQ.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.  
root@localhost's password:  
sending incremental file list  
  
sent 14,892 bytes received 2,350 bytes 3,134.91 bytes/sec  
total size is 1,330,517 speedup is 77.17  
aso@aso-client:~$ |
```

2. Ara creem fitxer en “/root” > Fem rsync > Eliminem el fitxer > Fem rsync → Podrem veure que en la còpia continua existint.

Si volem que sigui una sincronització exacta haurem d’especificar el paràmetre “--delete” i eliminarà del destí aquells fitxers que no estiguin en l’origen.

NOTA: Veiem que la còpia de “rsync” és molt menor i més ràpida.

“sudo touch /root/hola”

“sudo rsync -avz /root -e ssh root@localhost:/backup/rsync-backup”

“sudo rm /root/hola”

“sudo rsync -avz /root --delete -e ssh root@localhost:/backup/rsync-backup”

3. Si volem treure els fitxers “.txt”.

“sudo rsync -avz /root --exclude=*.txt -e ssh root@localhost:/backup/rsync-backup”

És important notar que:

“/src /dest” → Carpeta (literal) “src” dins de “dest”.

“/src/ dest” → Contingut de “src” dins de “dest”.

El “/” en el destí no afecta en res.

Making reverse incremental backups

Per aquesta part primer observem que es fa una còpia total i que, a mesura que anem eliminant/creant/modificant fitxers de l’origen i executant el script → Només es van fent còpies incrementals.

```
aso@aso-client:~$ sudo ls -la /backup/rsync-backup/complet/root
total 28
drwx----- 1 root root 160 Dec  2 07:19 .
drwx----- 1 root root   8 Dec  2 07:17 ..
-rw----- 1 root root 2068 Dec  2 06:48 .bash_history
-rw-r--r-- 1 root root 3176 Jan 30 2022 .bashrc
drwx----- 1 root root   74 Oct  2 12:45 .cache
-rw----- 1 root root   63 Nov 30 08:49 .lessht
drwxr-xr-x 1 root root  10 Nov 13 14:14 .local
-rw-r--r-- 1 root root   66 Nov 13 14:14 .selected_editor
drwx----- 1 root root   52 Dec  2 06:51 .ssh
drwxr-xr-x 1 root root  20 Sep 25 12:42 .vim
-rw----- 1 root root 9317 Dec  2 06:47 .viminfo
-rw-r--r-- 1 root root    0 Dec  2 07:18 adeu
-rw-r--r-- 1 root root    0 Dec  2 07:19 hola
aso@aso-client:~$ sudo rm /root/adeu
aso@aso-client:~$ sudo ./sync.sh
root@localhost's password:
sending incremental file list
deleting root/adeu
root/
sent 3,698 bytes received 91 bytes 1,082.57 bytes/sec
total size is 1,330,517 speedup is 351.15
aso@aso-client:~$ sudo ls -la /backup/rsync-backup/complet/root
total 28
drwx----- 1 root root 152 Dec  2 07:19 .
drwx----- 1 root root   8 Dec  2 07:17 ..
-rw----- 1 root root 2068 Dec  2 06:48 .bash_history
-rw-r--r-- 1 root root 3176 Jan 30 2022 .bashrc
drwx----- 1 root root   74 Oct  2 12:45 .cache
-rw----- 1 root root   63 Nov 30 08:49 .lessht
drwxr-xr-x 1 root root  10 Nov 13 14:14 .local
-rw-r--r-- 1 root root   66 Nov 13 14:14 .selected_editor
drwx----- 1 root root   52 Dec  2 06:51 .ssh
drwxr-xr-x 1 root root  20 Sep 25 12:42 .vim
-rw----- 1 root root 9317 Dec  2 06:47 .viminfo
-rw-r--r-- 1 root root    0 Dec  2 07:19 hola ?
aso@aso-client:~$ |
```

Snapshot-Style Backups

Review of hard links

“**cp --remove-destination og new**” elimina el new (si existia) i el crea de nou amb el contingut de l’original. Això implica que es crea un nou inode per al destí.

“**cp -l og new**” crea un hard-link.

Script to make snapshot backups

IMPORTANT: No hi ha servidor SFTP així que faig servir aquest script

```
#!/bin/bash

# author: @impulsado

# GLOBAL VARIABLES
SRC="/root"
SNAP0="/backup/backup.0"
SNAP1="/backup/backup.1"
SNAP2="/backup/backup.2"
SNAP3="/backup/backup.3"

# BASE CASE
if [ "$(id -u)" -ne 0 ]; then
    echo "ERROR: Must be root" >&2
    exit 1
fi

if [ ! -d "$SRC" ]; then
    echo "ERROR: $SRC must exists" >&2
    exit 1
fi

# GENERAL CASE
# 1. Delete the oldest backup
if [ -d "$SNAP3" ]; then
    rm -rf "$SNAP3"
fi

# 2. backup.2 --> backup.3
if [ -d "$SNAP2" ]; then
    mv "$SNAP2" "$SNAP3"
fi

# 3. backup.1 --> backup.2
if [ -d "$SNAP1" ]; then
    mv "$SNAP1" "$SNAP2"
fi
```

```
# 4. backup.0 --> backup.1
if [ -d "$SNAP0" ]; then
    mv "$SNAP0" "$SNAP1"
fi

# Create directory for the actual backup
mkdir -p "$SNAP0"

if [ -d "$SNAP1" ]; then
    # Use previous snapshot as a hard-link
    rsync -a -v --delete --link-dest="$SNAP1" "$SRC" "$SNAP0/"
else
    # There's no previous --> Crate the full backup
    rsync -a -v --delete "$SRC" "$SNAP0/"
fi
```

El que fa aquesta opció “**--link-dest**” és fer que: Si un fitxer no ha canviat respecte a l'especificat, en el nou backup hi haurà un hard-link en comptes d'un fitxer nou. D'aquesta forma estalviem espai.

Per a simular una pèrdua de dades i recuperació podem seguir aquestes comandes.

“**sudo mv /root /root.old**”

“**mkdir /root**”

“**chmod 700 /root**”

“**sudo chown root:root /root**”

“**sudo rsync -a -v /backup/rsync-backup/root/ /root**” ! MOLT IMPORANT la “/” de “src”

“**sudo ls -la /root**” → Hauria d'estar tot OK.