Backups

Backup… in simple terms

- Having a regular fresh copy of important files on the system, which can be restored either fully or partially
- Backup can be as simple as copying files using native Linux commands like `cp`, `tar`, `rsync`, or `dump`. It can be as complex as using a third party *backup solutions*. Those can be free like Bacula, Amanda, and Mondo, or paid like NetBackup.
- The destination to which files get copied (backed up) is called *backup media*. This can be a disk, an optical disk, or a tape depending on several factors like cost and required *data retention*.
- The whole operation is governed by a *backup policy*, which entails the following:
  - The amount of data that needs to be backed up. This dictates the amount and type of needed storage.
  - The frequency by which files change. This lets you know how often backup is needed (hourly, daily, weekly…etc.)
  - The duration for which data needs to be kept (data retention). This, too, controls the amount and type of backup media.

LAB: Calculate the amount of data that needs to be backed up

- Depending on your environment, you'll probably have a number of machines on your network that needs backup. In this lab you write a script that will calculate the total amount of space and the used one on a number of systems:

```
#!/bin/bash
TOTAL=0
USED=0
OUT=""
MACHINES=( 192.168.0.103 192.168.0.101 )
for m in "${MACHINES[@]}"
      do
            OUT="$OUT~`ssh  $m 'df / | tail -n +2'`"
      done
LENGTH=${#MACHINES[@]}
LENGTH=$((LENGTH + 1))
for (( i=2; i<=$LENGTH; i++ ))
      do
            USED_TEMP=`echo $OUT | cut -d "~" -f $i | cut -d " " -f 3`
            TOTAL_TEMP=`echo $OUT | cut -d "~" -f $i | cut -d " " -f 2`
            USED=$((USED+USED_TEMP))
            TOTAL=$((TOTAL+TOTAL_TEMP))
      done
TOTAL=$((TOTAL/1024/1024))
USED=$((USED / 1024 / 1024))
echo "Total is " $TOTAL " GB"
echo "Total used is " $USED " GB"
```

LAB: Calculate the frequency by which files change

- In this lab, we are going to write a script that sums up the sizes of files that were changed in the last 24 hours. The results are mailed to root. Such a script can be added to a cron job that will carry it out periodically.

```
#!/bin/bash
SUM=0
FILES=`find / -mount -type f -mtime -1 | xargs du -k | awk
'{print $1}'`
for i in $FILES; do
   SUM=$((SUM+i))
done
SUM=$((SUM/1024))
echo "$SUM MB" | mail -s "$HOSTNAME free space" root
```

Using nc for non-SSL-based remote backups

- Short for netcat, a utility that is used to make a lot of network-related operations. We are using it here as a daemon to aid in making remote backups.
- Data is transferred unencrypted so nc should not be used to backup sensitive data
- On the destination server, we used netcat to receive files by choosing a port and letting nc wait for input
  ```
  nc -l 20000| tar -xvf -
  ```
- Tar can be replaced with any other command that can read from standard input (like dump)
- On the client machine, we can issue a command like this
  ```
  tar -cvf - file(s) | nc destination_ip 20000
  ```

Using `cp`

- It is the simplest native tool used for backup
- It can be used to copy files and directories to a local or a remote directory.
- Typical usage is as follows:
  ```
  cp -ax source destination
  ```
  where `a` is *archive mode,* which preserves permissions and other file attributes. The x switch ensures that the command does not leave the filesystem to other mounted filesystems
- To include directories
  ```
  cp -Rax source destination
  ```
- Shortcomings of using `cp`:
  - Does not support resuming broken operations
  - Does not use SSL (when copying to remote directories)
  - The remote destination directory must be shared through SAMBA or NFS, which provides an extra burden
  - Does not support differential backups
- Used for very basic backup operations

Using tar

- Originally designed for tape backups (short for Tape Archive)
- Can still be used to backup to tapes
```
tar -cf /dev/rmt0 /root
```
- Can also be used to backup to a regular file (called tarball)
```
tar -cf /backup/mybackup.tar /root
```
- Commonly used with gzip to compress tarballs
```
gzip /backup/mybackup.tar
```
- Use the x argument to restore a number of files or the whole archive
```
tar -xf /backup/mybackup.tar [file(s)]
```
- It can be used with remote directories and it supports SSL (using SSH)
```
tar -cvf - file(s) | (ssh remote_server 'tar -C directory -
xvf -')
```
- The tar method has the following shortcomings:
  - Does not support resuming broken operations
  - Does not support incremental/differential backup

Using rsync

- A robust command used to copy files locally or to a remote server
- Supports SSL natively
- Can resume broken operations
- Supports differential/incremental backups because only new or changed files get copied
- Has options to synchronize directories. This is very useful when uploading files that change a lot in mirrored locations like webserver production and development machines
- When used in backup, it is most commonly used with the -a command switch (for archive), which preserves file permissions, symbolic links, and other file attributes
- Can be used with –z to compress files while being copied. This slightly speeds up backups on networks but consumes CPU
- Can be used with –P to provide a progress bar.

Using dump

- Sometimes it is considered the standard backup tool for Linux/UNIX
- It provides ten levels of backup (0-9), where 0 is full backup, every integer above 0 makes dump examine any new or changed files since that last integer backup. For example level 5 will take all changed/added files since level 4.
- It can be used to backup to tapes. When the tape is full, it will continue the operation when the next tape is ready.
- It can also be used to backup to a file, and – combined with SSH – can be used to backup to remote directories
- The typical usage for backup
  ```
  dump 0f  /dev/rmt0 /filesystem
  ```
  where `0` is the level, `/dev/rmt0` is the backup media (can be a regular file), and `/filesystem` is the path to the filesytem or directory to be backed up
- When restoring (must be inside the directory where you want to restore)
  ```
  restore rf /path/to/mybackup.dmp
  ```

LAB: cloning an entire Linux machine using dump and dd

- This method is used to backup an entire machine to a disk so that it can be used to boot to an identical copy, or to restore the original machine in case of disaster. The following procedure does <u>not</u> assume that the two machines are having identical disk sizes
- The operation can be summarized as follows:
  - [Source Machine]: Turn off and boot from the installation DVD, enter ”rescue mode”. You can skip this step if are absolutely sure no files will be changed during the cloning process
  - [Destination machine]
    - Start the destination machine using the installation DVD and enter “rescue mode”
    - Choose to start network manager
    - Choose not to mount the disks

- Clone the superblock from the source machine to the destination one using dd and nc:
  - [Destintation machine] : `nc -l 20000 | dd of=/dev/sda`
  - [Source machine]: `dd if=/dev/sda bs=446 count=1 | nc 192.168.0.105 20000`
  - We used 446 because it contains the bootstrap information. The remaining bytes are 64 for partition table (we created a different partition table), and 2 for signature totaling 512 (boot block).
- [Destination machine]:
  - Using fdisk, create a primary partition for boot
  - Create a new partition for the volume group (assuming the root volume group is using only one physical volume)
  - Create a volume group with the <u>same name</u> as the one on the source machine
  - Create all the required logical volumes that were on the source machine. Sizes do not have to be the same
  - Don’t forget to crate an lv for the SWAP space.
  - Using mkfs.ext4, create new filesystems on the new logical volumes. Use mkswap to format the swap space.
  - Mount the logical volumes on temporary directories

- Start cloning files from source to destination machines:
  - [Destination machine while in `/mnt/slash`]
    ```
    nc -l 20000 | restore -rf -
    ```
  - [Source machine] dump 0f - /dev/vg_redhat01/lv_root | nc 192.168.0.5 20000
- [Destination machine] After all files are cloned, examine the fstab file and correct any problems that might prevent the system from booting. For example having the original UUID of /dev/sda1, which has to be changed or replaced directly with /dev/sda
- [Destination machine] reboot the machine using the installation DVD and enter rescue mode. This time you can opt to let it mount the disks for you under /mnt/sysimage.
- [Destination machine] make /mnt/sysimage your root filesystem using chroot /mnt/sysimage
- [Destination machine] install GRUB using the following commands:

  -
    ```
    $ grub
    grub> root (hd0,0)
    grub> setup (hd0)
    grub> quit
    ```
- Reboot the system and ensure that it is running as expected