Backup and (hopefully) Restore

Andrea Gussoni

P.O.u.L.

23 Marzo 2017

POLITECNICO OPEN
unix LABS
Come hack with us.

Why do we need backups?

Bad things can happen and do happen:

- You may drop your computer accidentally.
- The disk may be damaged by vibrations during the daily commute.
- The computer where you keep the unique copy of your thesis may be stolen.
- After some time the disk may simply stop operating because of ageing.
- But often the principal cause of data loss is that thing that it is between the keyboard and the chair.

Why do we need backups?

[0] https://twitter.com/gitlabstatus/status/826591961444384768

What are backups?

Definition

The copying and archiving of computer data so that it may be used to restore the original after a data loss event.

What to backup?

It is important to distinguish what it is necessary to backup from what it is not.

What to backup?

It is important to distinguish what it is necessary to backup from what it is not.
Obviously this depends on the setup that you are using (native services, containers, VMs etc...)

A general guideline

Must:

- /home

At your discretion:

- /etc
- /var
- /mnt /media

Not necessary[1]:

- /proc /sys
- /dev /tmp

[1] if these folders contain something important probably you are doing something wrong in your setup

Backup types

Backups can be:

- **full**: a complete backup of a all files and folder starting from a root node.
- **incremental**: contains all the differences since the last incremental backup.
- **differential** contains the changes since the last full backup.

Backup Support

- Hard disks (HDD).
- Solid-State drives (SSD).
- Optical supports: DVDs, Blu-ray.
- Flash Drives.
- Cloud[2].

[2] Remember that there is no cloud, just other people's computers.

dd

dd is a powerful tool that basically can copy everything that is a file or a block device. It is common to use it for disk cloning. Usage example:

- *dd if=/dev/sdX of=/dev/sdY conv=fdatasync*[3]
  - **if:** input file/device
  - **out:** output file/device

[3] useful to actually wait the end of data transfer and avoid corrupted copies

dd

dd is a powerful tool that basically can copy everything that is a file or a block device. It is common to use it for disk cloning. Usage example:

- *dd if=/dev/sdX of=/dev/sdY conv=fdatasync*[3]
  - **if:** input file/device
  - **out:** output file/device

Caution

Since **dd** often requires *sudo* privileges to run, if you mismatch the name of a device you can actually wipe the content of your primary hard disk, double check always the arguments before pressing enter.

[3] useful to actually wait the end of data transfer and avoid corrupted copies

GNU ddrescue

gdrescue is an enhanced version of dd that tries to rescue good parts in case of read errors. It may be useful to recover data from a drive with some damaged sector.
Usage Example:

- *ddrescue [options] /dev/sdX outfile mapfile*
  - **mapfile:** a human readable text file ddrescue uses to manage the copy

GNU ddrescue

gdrescue is an enhanced version of dd that tries to rescue good parts in case of read errors. It may be useful to recover data from a drive with some damaged sector.
Usage Example:

- *ddrescue [options] /dev/sdX outfile mapfile*
    - **mapfile:** a human readable text file ddrescue uses to manage the copy

Caution

For the rescued data to be correct, both dd and gddrescue are best used on unmounted devices.

GNU ddrescue

gdrescue is an enhanced version of dd that tries to rescue good parts in case of read errors. It may be useful to recover data from a drive with some damaged sector.
Usage Example:

- *ddrescue [options] /dev/sdX outfile mapfile*
  - **mapfile:** a human readable text file ddrescue uses to manage the copy

Caution

For the rescued data to be correct, both dd and gddrescue are best used on unmounted devices.

Tip

gddrescue can also be useful when trying to reallocate sectors on a drive with a few sector unreadable. Doing a wipe of the drive with gddrescue should reallocate bad sectors.

rsync

Also known as an advanced version of cp

Pros

- (unlike cp) preserves links, file permissions and ownerships, modification times, etc.
- designed to be network efficient because only transfers file changes.
- easy to use.

Cons

- no storage encryption.

rsync: usage

- rsync -Pr source destination
  - **P:** keep partially transferred files if the transfer is interrupted.
  - **r:** recursive directory option.
  - this do not preserve the attributes of the file.

[4] But please don't do this *rsync -av --delete source host:~*

rsync: usage

- rsync -Pr source destination
  - **P:** keep partially transferred files if the transfer is interrupted.
  - **r:** recursive directory option.
  - this do not preserve the attributes of the file.

- rsync source host:destination[4]
  - uses ssh by default, but can also be forced with the -e ssh option.

[4] But please don't do this *rsync -av --delete source host:~*

rsync: usage

- rsync -Pr source destination
  - **P:** keep partially transferred files if the transfer is interrupted.
  - **r:** recursive directory option.
  - this do not preserve the attributes of the file.

- rsync source host:destination[4]
  - uses ssh by default, but can also be forced with the -e ssh option.

- rsync -aAXv --exclude={...} /* /backupfolder
  - backup /* while following symlinks and preserving file properties.

[4] But please don't do this *rsync -av --delete source host:~*

rsnapshot: rsync automated

rsnapshot produces automated, periodical system snapshots

Pros

- preserves links, file permissions and ownership, modification times, etc.
- network efficient.
- each snapshot contains a full system backup.
- easy to use.

Cons

- no storage encryption.

duplicity

duplicity produces encrypted, incremental backups in tar format.

Pros

- preserves links, file permissions and ownership, modification times, etc.
- network efficient.
- incremental backups.
- supports storage encryption with gpg.
- easy to use.

duplicity: usage

- duplicity /home/user scp::/user@host//backup/directory

duplicity: usage

- duplicity /home/user scp::/user@host//backup/directory

- duplicity [restore] scp://user@host//backup/directory /home/user

duplicity: usage

- duplicity /home/user scp::/user@host//backup/directory
- duplicity [restore] scp://user@host//backup/directory /home/user
- duplicity full /home/user scp::/user@host//backup/directory

duplicity: usage

- duplicity list-current-files scp::/user@host//backup/directory
  - list the files contained in the backup.

duplicity: usage

- duplicity list-current-files scp::/user@host//backup/directory
  - list the files contained in the backup.

- duplicity [restore] -t 3D scp://user@host//backup/directory /home/user
  - specify the time from which to restore files.

duplicity: usage

- duplicity list-current-files scp::/user@host//backup/directory
  - list the files contained in the backup.
- duplicity [restore] -t 3D scp://user@host//backup/directory /home/user
  - specify the time from which to restore files.
- duplicity remove-older-than 30D scp::/user@host//backup/directory
  - remove from the backup full backups older than the specified period.

Demo

Demo!

Last but not Least

- When you use duplicity with encryption enabled always remember to backup the gpg keys you use to encrypt and sign the backup.
  If you loose them you won't be able to restore the backup.

Last but not Least

- When you use duplicity with encryption enabled always remember to backup the gpg keys you use to encrypt and sign the backup.
  If you loose them you won't be able to restore the backup.
- Always check that the backup is taking place, don't just assume that everything is working fine because you followed exactly the suggested guide.

Last but not Least

- When you use duplicity with encryption enabled always remember to backup the gpg keys you use to encrypt and sign the backup.
  If you loose them you won't be able to restore the backup.
- Always check that the backup is taking place, don't just assume that everything is working fine because you followed exactly the suggested guide.
- Always try to test that the backup is really working by trying to restore the backup. You'll be surprised to know how many times the backup procedures are not really working, and unfortunately if you do not test them you'll notice it only when the files are gone.

Hi again GitLab

Problems Encountered

1. LVM snapshots are by default only taken once every 24 hours. YP happened to run one manually about 6 hours prior to the outage
2. Regular backups seem to also only be taken once per 24 hours, though YP has not yet been able to figure out where they are stored. According to JN these don't appear to be working, producing files only a few bytes in size.
    a. SH: It looks like pg_dump may be failing because PostgreSQL 9.2 binaries are being run instead of 9.6 binaries. This happens because omnibus only uses Pg 9.6 if data/PG_VERSION is set to 9.6, but on workers this file does not exist. As a result it defaults to 9.2, failing silently. No SQL dumps were made as a result. Fog gem may have cleaned out older backups.
3. Disk snapshots in Azure are enabled for the NFS server, but not for the DB servers.
4. The synchronisation process removes webhooks once it has synchronised data to staging. Unless we can pull these from a regular backup from the past 24 hours they will be lost
5. The replication procedure is super fragile, prone to error, relies on a handful of random shell scripts, and is badly documented
    a. SH: We learned later the staging DB refresh works by taking a snapshot of the gitlab_replicator directory, prunes the replication configuration, and starts up a separate PostgreSQL server.
6. Our backups to S3 apparently don't work either: the bucket is empty
7. We don't have solid alerting/paging for when backups fails, we are seeing this in the dev host too now.

So in other words, out of 5 backup/replication techniques deployed none are working reliably or set up in the first place. => we're now restoring a backup from 6 hours ago that worked

[4] https://docs.google.com/document/d/1GCK53YDcBWQveod9kfzW-VCxIABGiryG7_z_6jHdVik/pub

Before the Backup

A different approach to data protection is to use RAID (*Redundant Array of Independent Disks*).

[4] For further informations you can visit `https://www.digitalocean.com/community/tutorials/an-introduction-to-raid-terminology-and-concepts`

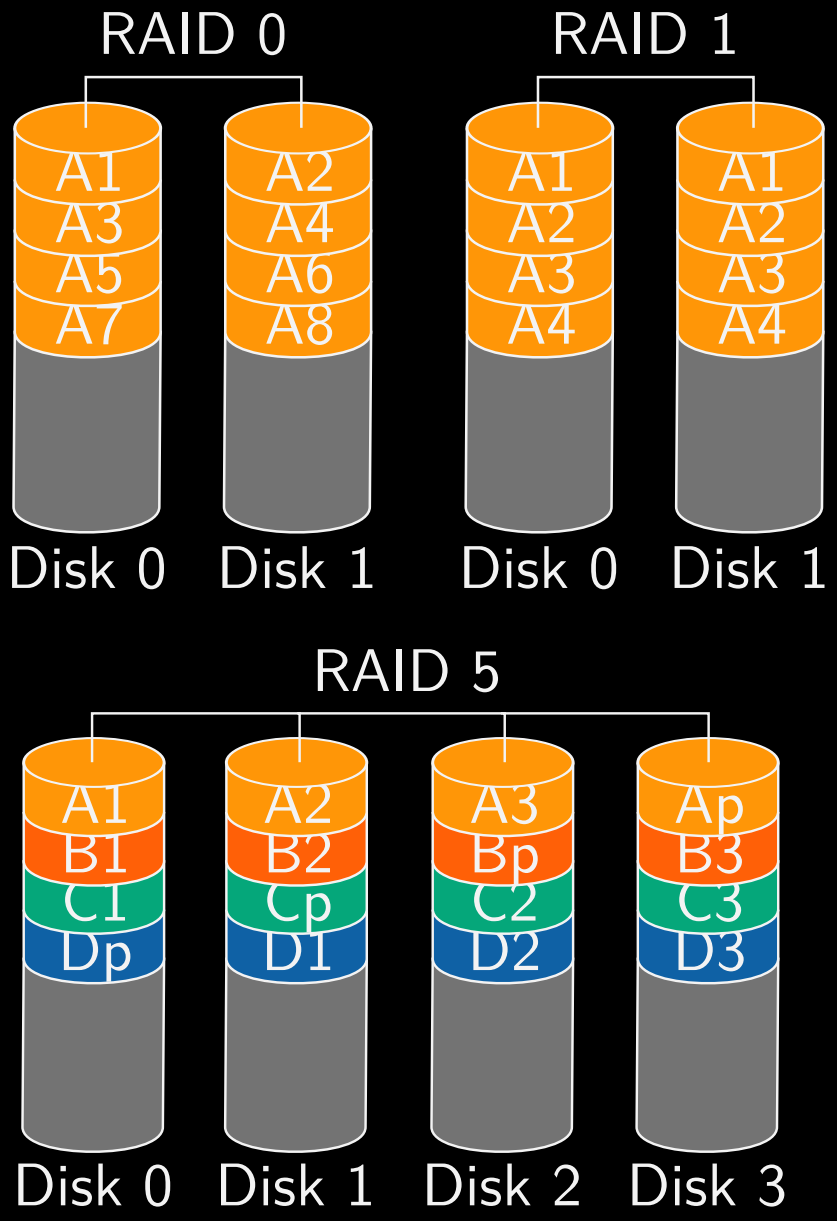
Before the Backup

A different approach to data protection is to use RAID (*Redundant Array of Independent Disks*).
In general what we try to obtain with RAID is:

- Survival of the system if a disk failure happen.
- In certain conditions we can achieve higher performances compared to the single disk case.

[4] For further informations you can visit `https://www.digitalocean.com/community/tutorials/an-introduction-to-raid-terminology-and-concepts`

RAID Configurations



New generation filesystems

There are new kind of filesystems that try to resolve some problems that we usually have in data storage. The two main examples are ZFS and Btrfs[5] Classical features that we can find in this kind of filesystems are:

- CopyOnWrite.
- Deduplication.
- Data & Metadata checksums.
- Integrated RAID.
- Volume Management.
- Snapshots.

[5] Please remind that Btrfs is still in heavy development, before using it in production check at `https://btrfs.wiki.kernel.org/index.php/Status` that the features you will need are considered stable.

Snapshots

- Snapshots can be particularly useful because they allow us to obtain an (almost) instant snapshot of a volume that we can restore later, archive somewhere etc.

Snapshots

- Snapshots can be particularly useful because they allow us to obtain an (almost) instant snapshot of a volume that we can restore later, archive somewhere etc.
- So we can use them in order to do some potential risky modifications on a system and restore the previous state with a little effort.

Snapshots

- Snapshots can be particularly useful because they allow us to obtain an (almost) instant snapshot of a volume that we can restore later, archive somewhere etc.
- So we can use them in order to do some potential risky modifications on a system and restore the previous state with a little effort.
- Remember that having a separate *classical* backup is always useful, in particular for important data of our applications.

Snapshots

- Snapshots can be particularly useful because they allow us to obtain an (almost) instant snapshot of a volume that we can restore later, archive somewhere etc.
- So we can use them in order to do some potential risky modifications on a system and restore the previous state with a little effort.
- Remember that having a separate *classical* backup is always useful, in particular for important data of our applications.
- RAID is not a backup.

References

- https://wiki.archlinux.org/index.php/Full_system_backup_with_rsync
- https://wiki.archlinux.org/index.php/Duplicity
- http://duplicity.nongnu.org/
- https://www.digitalocean.com/community/tutorials/how-to-use-duplicity-with-gpg-to-securely-automate-backups-on-ubuntu
- https://github.com/zertrin/duplicity-backup.sh
- https://wiki.archlinux.org/index.php/Rsnapshot
- https://slides.poul.org/2017/corsi-linux-avanzati/backup_handbook.pdf

Special Thanks

I used as reference and starting point for this presentation the material of the previous editions of the course.
Special thanks to *Valeria Mazzola*[6] and *Federico Amedeo Izzo*[7] for the slides of the two previous edition of this talk.

---

[6] https://slides.poul.org/2016/corsi-linux-avanzati/Backup_and_Restore.pdf

[7] https://filesystem.izzo.ovh/

License

Thank you!