LVM SNAPSHOT

What is an LVM Snapshot?

An **LVM snapshot** is a **point-in-time, virtual copy** of a logical volume (LV) created using **copy-on-write (CoW)** technology. It captures the exact state of the volume at the moment of creation and remains consistent even as the original volume continues to be modified.

How It Works

1. When you create a **snapshot**, it initially shares all data blocks with the origin LV.
2. As the origin is updated, the original data blocks are **copied to the snapshot volume** before the changes occur—ensuring the snapshot reflects the original state.
3. Only changed blocks are stored in the snapshot, making it **space-efficient**
4. New blocks (never before written) aren't copied—only overwritten blocks trigger CoW.

Why Use Snapshots?

Consistent Backups with Zero Downtime

Ideal for backing up live systems—snapshots freeze a stable state without disrupting ongoing operations

Testing & Safe Rollbacks

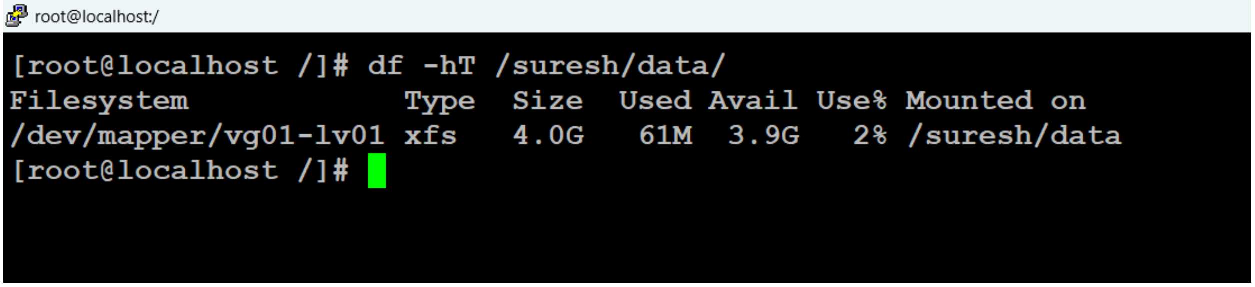
Perfect for applying patches, upgrades, or experiments on a separate copy. If issues arise, you can rollback effortlessly

IMPORTANT NOTE:

In LVM we have a feature where an increasing snapshot size **only prevents early fill-up**; it doesn’t retroactively include data changed before extension. Only blocks changed *after* creation are tracked and restorable.

Workflow Example

Step 1: Check the filesystem(/suresh/data) usage

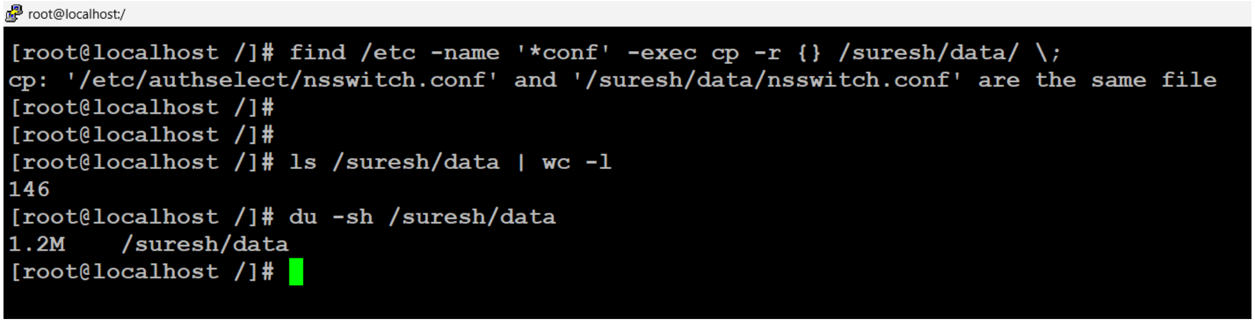


```
[root@localhost /]# df -hT /suresh/data/
Filesystem              Type  Size  Used Avail Use% Mounted on
/dev/mapper/vg01-lv01 xfs   4.0G   61M  3.9G   2% /suresh/data
[root@localhost /]#
```

This reveals how full the **/suresh/data** filesystem is (e.g., total size, used, available, percentage).
Here, **lv is lv01** and **vg is vg01**

Step 2: Add data/files in /suresh/data

Command: find /etc -name ‘*.conf’ -exec cp -r {} /suresh/data/ \;



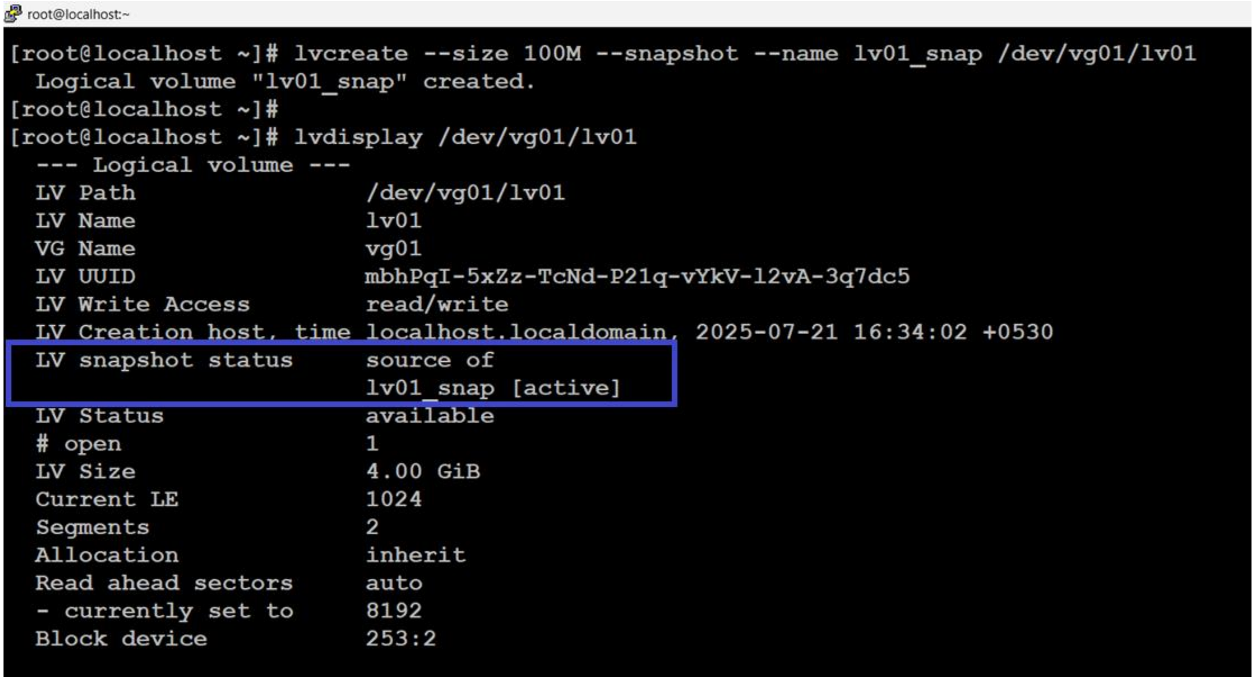
```
[root@localhost /]# find /etc -name '*conf' -exec cp -r {} /suresh/data/ \;
cp: '/etc/authselect/nsswitch.conf' and '/suresh/data/nsswitch.conf' are the same file
[root@localhost /]#
[root@localhost /]#
[root@localhost /]# ls /suresh/data | wc -l
146
[root@localhost /]# du -sh /suresh/data
1.2M    /suresh/data
[root@localhost /]#
```

Step 3: Create LV Snapshot

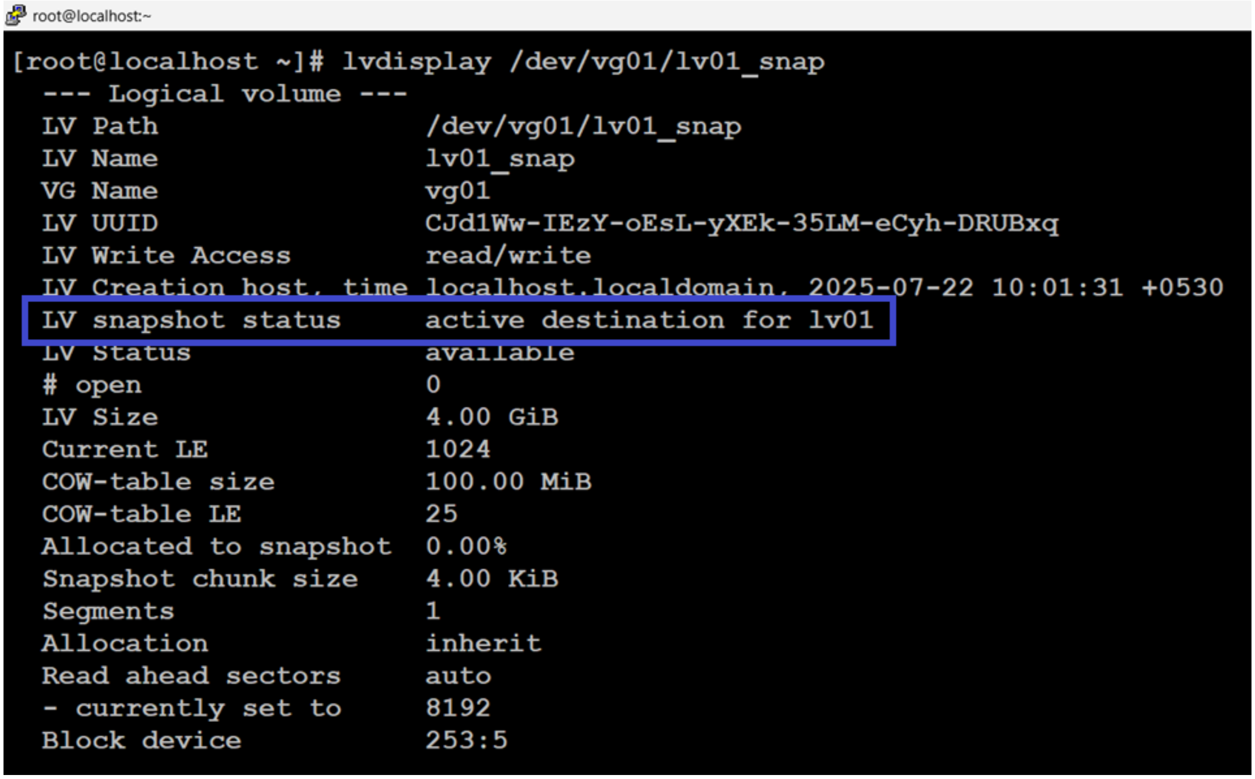
Command:

lvcreate -L 1.5G -s -n lv01_snap /dev/vg01/lv01

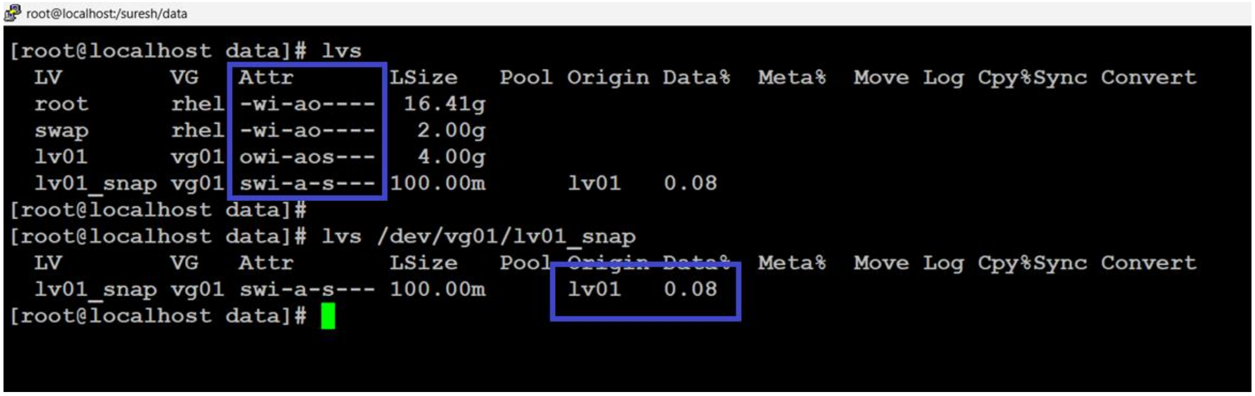
lvcreate --size 1.5G --snapshot --name lv01_snap /dev/vg01/lv01



```
[root@localhost ~]# lvcreate --size 100M --snapshot --name lv01_snap /dev/vg01/lv01
  Logical volume "lv01_snap" created.
[root@localhost ~]#
[root@localhost ~]# lvdisplay /dev/vg01/lv01
  --- Logical volume ---
  LV Path                /dev/vg01/lv01
  LV Name                lv01
  VG Name                vg01
  LV UUID                mbhPqI-5xZz-TcNd-P21q-vYkV-12vA-3q7dc5
  LV Write Access        read/write
  LV Creation host, time localhost.localdomain, 2025-07-21 16:34:02 +0530
  LV snapshot status     source of
                         lv01_snap [active]
  LV Status              available
  # open                 1
  LV Size                4.00 GiB
  Current LE             1024
  Segments               2
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:2
```



```
[root@localhost ~]# lvdisplay /dev/vg01/lv01_snap
  --- Logical volume ---
  LV Path                /dev/vg01/lv01_snap
  LV Name                lv01_snap
  VG Name                vg01
  LV UUID                CJd1Ww-IEzY-oEsL-yXEk-35LM-eCyh-DRUBxq
  LV Write Access        read/write
  LV Creation host, time localhost.localdomain, 2025-07-22 10:01:31 +0530
  LV snapshot status     active destination for lv01
  LV Status              available
  # open                 0
  LV Size                4.00 GiB
  Current LE             1024
  COW-table size         100.00 MiB
  COW-table LE           25
  Allocated to snapshot  0.00%
  Snapshot chunk size    4.00 KiB
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:5
```



```
[root@localhost data]# lvs
  LV        VG   Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  root      rhel -wi-ao----  16.41g
  swap      rhel -wi-ao----   2.00g
  lv01      vg01 owi-aos---   4.00g
  lv01_snap vg01 swi-a-s--- 100.00m      lv01   0.08
[root@localhost data]#
[root@localhost data]# lvs /dev/vg01/lv01_snap
  LV        VG   Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  lv01_snap vg01 swi-a-s--- 100.00m      lv01   0.08
[root@localhost data]#
```

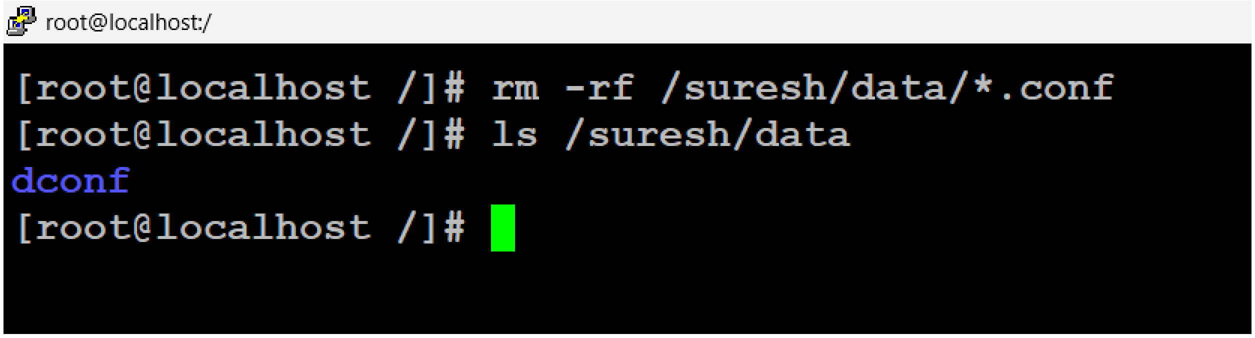
- **lv01_snap** is a thin snapshot of **lv01**—thin snapshots consume storage only as the original volume changes.

- At **0.08% data usage**, almost no data has changed since snapshot creation: indicating little write activity or early snapshot state.

- The **100 MB LSize** is the allocated snapshot space; given the low Data%, this is sufficient—but once changes exceed ~100 MB, the snapshot may fail due to space exhaustion.

Step 4: Remove the data or unfortunately data will deleted in /suresh/data

Command: rm -rf /suresh/data/*.conf



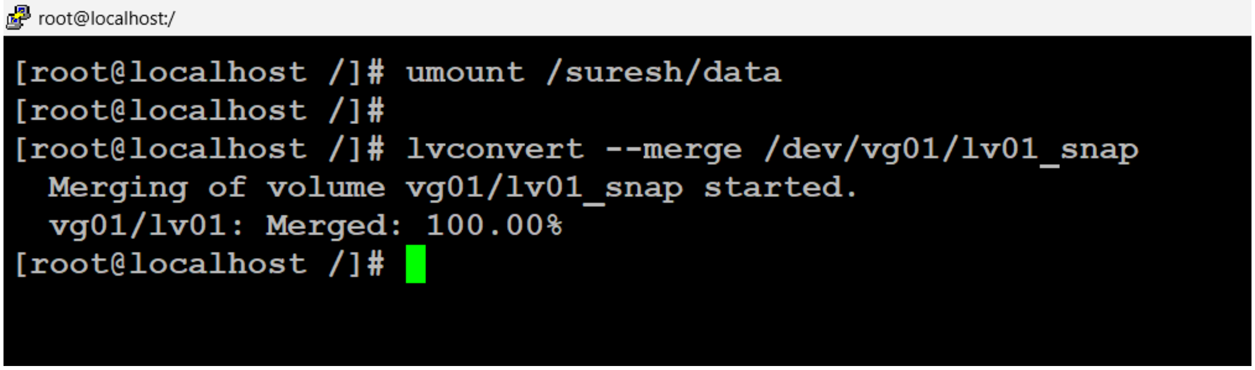
```
[root@localhost /]# rm -rf /suresh/data/*.conf
[root@localhost /]# ls /suresh/data
dconf
[root@localhost /]# 
```

Step 5: Unmount the LV and Start merge with lvconvert --merge

Command:

umount /suresh/data

lvconvert –merge /dev/vg01/lv01_snap



```
[root@localhost /]# umount /suresh/data
[root@localhost /]#
[root@localhost /]# lvconvert --merge /dev/vg01/lv01_snap
  Merging of volume vg01/lv01_snap started.
  vg01/lv01: Merged: 100.00%
[root@localhost /]#
```

This initiates merging the snapshot (**lv01_snap**) back into the origin LV (lv01).

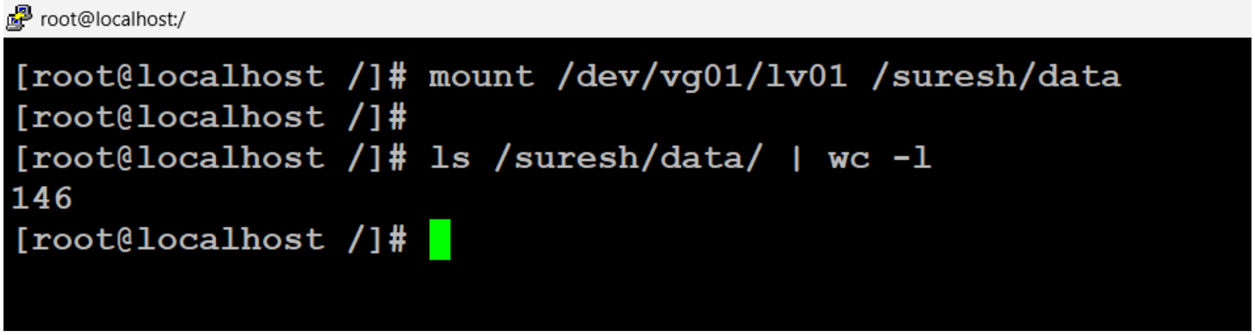
Unmount first: As done (**umount /suresh/data**), this allows merge to start immediately.

How **lvconvert --merge** works

- The command merges the snapshot's data into the origin LV.
- Both the snapshot and the origin LV must be closed/unmounted for immediate merge; otherwise, it's deferred until the next activation.
- After merge:
  - Snapshot LV is deleted.
  - Original LV retains its name, UUID, and metadata

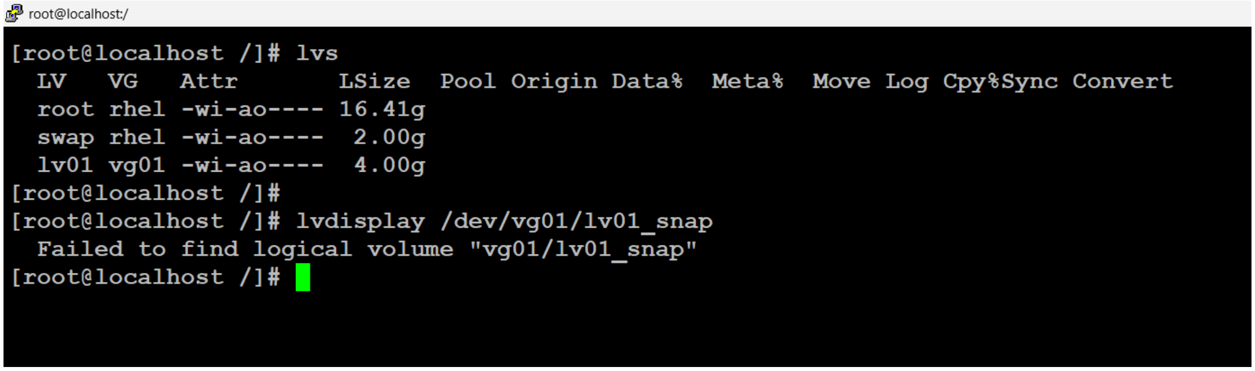
Step 6: mount the lv and check the data in /suresh/data

Command:



```
[root@localhost /]# mount /dev/vg01/lv01 /suresh/data
[root@localhost /]#
[root@localhost /]# ls /suresh/data/ | wc -l
146
[root@localhost /]#
```

- After merging the snapshot back into the origin (from your previous steps), you remounted **lv01** to verify the current contents.
- Seeing **146 items** confirms that:
  - The snapshot merge was successful.
  - The filesystem is intact and showing expected contents.
- If you know the baseline count from before the snapshot, you can compare to ensure **no data was lost or added unexpectedly.**



root@localhost:/

```
[root@localhost /]# lvs
  LV   VG   Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  root rhel -wi-ao---- 16.41g
  swap rhel -wi-ao----  2.00g
  lv01 vg01 -wi-ao----  4.00g
[root@localhost /]#
[root@localhost /]# lvdisplay /dev/vg01/lv01_snap
  Failed to find logical volume "vg01/lv01_snap"
[root@localhost /]#
```

Manual Snapshot Extension

Check current snapshot usage

Command:

lvs -o vg_name,lv_name,origin,data_percent,lv_size

This reports how much of the snapshot's allocated CoW space is used

Deactivate snapshot (if required)

If the snapshot is active, deactivate it before resizing:

Command:

```
lvchange -an /dev/vg01/lv01_snap
```

Extend snapshot size

Add more CoW area using:

```
lvextend --size +<AdditionalSize>G /dev/<vg>/<snapshot_lv>
```

e.g., `lvextend --size +1G /dev/vg01/lv01_snap`

Verify the extension

Command:

```
lvs -o lv_name,lv_size,data_percent
```

Ensure the snapshot size increased and data% adjusted accordingly

Conclusion & Key Takeaways

An **LVM snapshot** is a powerful, point-in-time copy of a logical volume leveraging copy-on-write (CoW) technology. It enables consistent backups and risk-free testing environments without disrupting live operations. However, snapshots are **not a replacement for backups**—they require proper space management and monitoring, and are best used as part of a broader backup and recovery strategy

Best Practices

- **Create snapshots during quiet times**, ideally after quiescing applications (e.g. databases) to ensure consistency.
- **Monitor snapshot usage (data%)** regularly with `lvs`, and resize before it nears capacity to avoid failure.
- **Use snapshots as backup staging areas**, not long-term solutions. Transfer data off-volume rapidly, then lvremove the snapshot.
- **Minimize snapshot count**, as each adds I/O overhead.

LVM snapshots are excellent tools for:

1. **Zero-downtime backups** — capture consistent volume state.
2. **Safe testing and rollback** — revert to known-good state easily.
3. **Efficient storage** — only track changes, not entire data.

But they must be **monitored for space**, **managed wisely**, and always used in tandem with a robust backup policy.