LVM thin provisioning

Thin pools, virtual capacity, snapshots, metadata and exhaustion

Logical versus physical capacity

A thin pool allocates physical blocks as thin volumes are written, rather than reserving all blocks when a volume is created. A 500 GiB thin LV may initially consume only a small fraction of that amount. This is useful for virtual machines, containers and sparse datasets, but the apparent virtual capacity can exceed the real free space in the pool.

Thin pools have both data and metadata areas. Data% indicates allocated data blocks; Meta% indicates metadata utilisation. Either area becoming exhausted can interrupt writes and may threaten data availability. Thin provisioning therefore requires active monitoring and capacity planning.

```
lvs -a -o lv_name,segtype,lv_size,data_percent,metadata_percent,lv_health_status
```

Creating a thin pool and thin volume

The example creates a 100 GiB thin pool and a 200 GiB thin LV. The thin volume's virtual size is not a promise that 200 GiB of physical capacity is available. Do not overcommit beyond a clearly understood growth and alerting plan.

After creation, format the thin LV and mount it normally. LVM handles allocation of physical blocks on first write. Discard/TRIM support and filesystem options affect whether freed blocks are returned to the pool.

```
sudo lvcreate -L 100G --type thin-pool -n pool0 vgdata
sudo lvcreate -V 200G -T vgdata/pool0 -n vmdata
sudo mkfs.ext4 /dev/vgdata/vmdata
lvs -a -o lv_name,lv_size,data_percent,metadata_percent
```

Monitoring and auto-extension

Inspect pool utilisation frequently, not just the virtual volume size. LVM supports thin_pool_autoextend_threshold and thin_pool_autoextend_percent in lvm.conf, normally coordinated with lvm2-monitor and dmeventd. Auto-extension only works if the VG has free extents; it cannot create disk capacity from nothing.

For alerting, define thresholds well before 100%, and include growth rate in the decision. A pool at 75% that grows 10% per hour is more urgent than one at 85% that is stable. Metadata growth can differ from data growth, especially with many snapshots.

```
sudo lvs -a -o lv_name,lv_size,data_percent,metadata_percent,seg_monitor
sudo vgs -o vg_name,vg_free
sudo lvextend -L +20G vgdata/pool0
```

Thin snapshots and discard

Thin snapshots share blocks with their origin until writes cause divergence. This can make snapshot creation fast and space efficient, but snapshots still consume pool capacity as data changes. A snapshot is not an independent backup and may become unusable if its thin pool fails.

Discard support can return unused filesystem blocks to the pool, depending on the storage stack. Test fstrim and inspect Data% before assuming it is effective. On layered storage, discards may be filtered by encryption, virtualization or device settings.

```
sudo lvcreate -s -n vmdata-before-change vgdata/vmdata
sudo fstrim -v /mountpoint
lvs -a -o lv_name,origin,pool_lv,data_percent
```

Failure scenarios and capacity policy

When a thin pool fills, applications may receive I/O errors or writes may be queued according to pool settings and version. Do not rely on indefinite queuing. Extend capacity promptly and investigate the condition before resuming a high-write workload.

A robust design reserves emergency free extents, monitors both data and metadata, limits snapshots, and ensures that the underlying VG and storage hardware are healthy. Keep recoverable backups outside the thin pool and document how to expand it.

```
lvs -a -o +lv_health_status,data_percent,metadata_percent
journalctl -k --since today | grep -Ei "thin|device-mapper|no space"
```

Reference scope: commands and defaults vary by distribution and software version. Consult the locally installed manual pages for exact syntax and behaviour.