LVM Logical volumes

START HERE

`lsblk`
Find the disk

`mkfs.ext4 /dev/<vg>/<lv>`
Format it

`pvcreate <dev>`
Make a PV

`lvs`
List LVs

`vgcreate <vg> <dev>`
Make a VG

`lvextend -r -L +<size> <vg>/<lv>`
Grow LV and fs

`lvcreate -L <size> -n <lv> <vg>`
Make an LV

`vgs`
Free space in VG

Disk / partition /dev/sdb1 → `pvcreate` → **Physical volume** PV → `vgcreate` → **Volume group** VG, pool of extents → `lvcreate` → **Logical volume** /dev/<vg>/<lv>

CONCEPTS

Disks become PVs, PVs pool into a VG, and you carve LVs out of the VG. An LV is a block device: put a filesystem on it.

Space is handed out in extents (4 MiB by default). Sizes are rounded to whole extents.

Name an LV as <vg>/<lv> or /dev/<vg>/<lv>. Use those paths, not /dev/mapper, in scripts.

Most commands need root. Add -t (test mode) to a change to try it first, where it is supported.

INSPECT

`pvs`
List physical volumes

`vgs`
List volume groups with free space

`lvs`
List logical volumes

`lvs -a -o +devices`
Show LVs and the PVs they use

`pvdisplay <pv>`
Detailed PV info

`vgdisplay <vg>`
Detailed VG info (extent size, free PE)

`lvdisplay <vg>/<lv>`
Detailed LV info and status

`lvs --units g`
Show sizes in a fixed unit

`lvs --noheadings --nosuffix --units b -o lv_size <vg>/<lv>`
Script-friendly size in bytes

`pvscan / vgscan / lvscan`
Rescan for PVs, VGs, LVs

`lvm fullreport`
PVs, VGs and LVs in one report

`lsblk`
Show the block device tree

PHYSICAL VOLUMES

`pvcreate <dev>...`
Initialise a disk or partition
Wipes the LVM label area; the device must not hold data you need.

PHYSICAL VOLUMES (cont.)

`pvremove <dev>`
Remove the LVM label from a PV

`pvresize <dev>`
Grow a PV after the disk got bigger

`pvmove <old-pv> [<new-pv>]`
Move all extents off a PV
Safe while the LVs are mounted. Run it again to resume after an interruption.

`pvmove -n <lv> <old-pv> <new-pv>`
Move only one LV's extents

VOLUME GROUPS

`vgcreate <vg> <pv>...`
Create a VG from PVs

`vgextend <vg> <pv>...`
Add PVs to a VG
Unused devices are initialised automatically; pvcreate first is optional.

`vgreduce <vg> <pv>`
Remove an empty PV from a VG
Free the PV first with pvmove.

`vgrename <old> <new>`
Rename a VG

`vgchange -ay <vg>`
Activate all LVs in a VG

`vgchange -an <vg>`
Deactivate all LVs in a VG

`vgremove <vg>`
Remove a VG (asks before removing LVs)

`vgsplit <vg> <new-vg> <pv>`
Split off PVs into a new VG

LOGICAL VOLUMES

`lvcreate -L <size> -n <lv> <vg>`
Create an LV of a fixed size

`lvcreate -l 100%FREE -n <lv> <vg>`
Create an LV from all free space

`lvcreate -l 50%VG -n <lv> <vg>`
Use a share of the whole VG

`lvcreate -L <size> -n <lv> <vg> <pv>`
Allocate only from one PV

`lvcreate --type raid1 -m 1 -L <size> -n <lv> <vg>`
Mirrored LV (RAID1)

LOGICAL VOLUMES (cont.)

`mkfs.ext4 /dev/<vg>/<lv>`
Put a filesystem on the LV

`mount /dev/<vg>/<lv> <dir>`
Mount the LV

`lvrename <vg> <old> <new>`
Rename an LV

`lvremove <vg>/<lv>`
Delete an LV (data is lost)

`lvchange -ay <vg>/<lv>`
Activate an LV

`lvchange -an <vg>/<lv>`
Deactivate an LV

`lvchange -pr <vg>/<lv>`
Make an LV read-only

RESIZE

`lvextend -L +<size> <vg>/<lv>`
Grow an LV by an amount

`lvextend -L <size> <vg>/<lv>`
Grow an LV to an absolute size

`lvextend -l +100%FREE <vg>/<lv>`
Grow an LV into all free space

`lvextend -r -L +<size> <vg>/<lv>`
Grow the LV and its filesystem

`lvresize -r -L -<size> <vg>/<lv>`
Shrink LV and filesystem together
-r shrinks the filesystem before the LV (it may unmount it). XFS cannot be shrunk, so this fails there.

`lvreduce -L -<size> <vg>/<lv>`
Shrink an LV only
Destroys data if the filesystem still uses the removed space.

`resize2fs /dev/<vg>/<lv>`
Grow an ext4 fs to fill its LV

`xfs_growfs <mountpoint>`
Grow a mounted XFS fs

Shrinking order: unmount, fsck, shrink the filesystem, then reduce the LV, never the other way round. Prefer lvresize -r and take a backup first.

Growing works online for ext4 and XFS. Extend the LV first, then the filesystem (or let -r do it).

SNAPSHOTS

`lvcreate -s -L <size> -n <snap> <vg>/<lv>`
Snapshot an LV
The size holds changed blocks only. A full snapshot gets invalidated.

`lvconvert --merge <vg>/<snap>`
Roll the origin back to a snapshot
If the origin or snapshot is open the merge is deferred until both are closed and one is next activated.

`lvremove <vg>/<snap>`
Drop a snapshot

`lvs -o +snap_percent`
Show how full a snapshot is

`lvextend -L +<size> <vg>/<snap>`
Give a snapshot more room

BACKUP AND REPAIR

`vgcfgbackup <vg>`
Back up VG metadata

`vgcfgrestore -l <vg>`
List metadata backups

`vgcfgrestore -f <file> <vg>`
Restore VG metadata

RECIPES

New data disk to mounted volume

```
pvcreate /dev/sdb
vgcreate data /dev/sdb
lvcreate -l 100%FREE -n files data
mkfs.ext4 /dev/data/files
mount /dev/data/files /srv/files
```

Add a disk and grow a mounted volume

```
pvcreate /dev/sdc
vgextend data /dev/sdc
lvextend -r -l +100%FREE data/files
df -h /srv/files
```

Replace a disk without downtime

```
pvcreate /dev/sdd
vgextend data /dev/sdd
pvmove /dev/sdb /dev/sdd
vgreduce data /dev/sdb
pvremove /dev/sdb
```

Snapshot, test, roll back

```
lvcreate -s -L 5G -n pre data/files
# ... make risky changes ...
umount /srv/files
lvconvert --merge data/pre
lvchange -an data/files &&
lvchange -ay data/files
mount /dev/data/files /srv/files
```