# Pengertian LVM (Logical Volume Manager)

LVM (Logical Volume Manager) adalah sistem manajemen penyimpanan dinamis di Linux yang memungkinkan penggunanya untuk mengelola ruang penyimpanan dengan fleksibel dibandingkan partisi tradisional. LVM memungkinkan penggabungan beberapa disk atau partisi menjadi satu volume logis yang dapat diperbesar atau diperkecil sesuai kebutuhan.

---

# Maksud & Fungsi LVM

## Maksud

LVM digunakan untuk mengatasi keterbatasan partisi statis dengan memberikan fleksibilitas dalam manajemen ruang penyimpanan.

## Fungsi LVM

1. Flexible Storage Management – Memungkinkan resize (menambah/mengurangi) ukuran partisi tanpa kehilangan data.
2. Multiple Disk Management – Menggabungkan beberapa disk fisik menjadi satu kesatuan (Volume Group).
3. Snapshot – Membuat salinan data secara instan untuk backup tanpa mengganggu sistem.
4. RAID-like Capabilities – Memungkinkan stripe atau mirror data antar disk untuk meningkatkan performa atau keamanan.
5. Easy Partitioning – Tidak terikat pada batasan partisi fisik seperti MBR/GPT.

---

# Cara Kerja LVM

LVM bekerja dengan menggunakan tiga komponen utama:

1. Physical Volume (PV)

- Merupakan perangkat fisik seperti HDD/SSD atau partisi yang digunakan sebagai storage untuk LVM.
- Dibuat menggunakan pvcreate.

3. Volume Group (VG)

- Kumpulan dari satu atau lebih PV yang membentuk satu kesatuan penyimpanan besar.
- Dibuat menggunakan vgcreate.

5. Logical Volume (LV)

- Unit penyimpanan yang digunakan oleh sistem operasi seperti layaknya partisi biasa.
- Dibuat menggunakan lvcreate.
- Dapat diperbesar (lvextend) atau diperkecil (lvreduce).

---

## Contoh Implementasi LVM di Linux

### 1. Membuat LVM dari Awal

Asumsi:

- Ada 2 disk baru (/dev/sdb dan /dev/sdc) yang akan digunakan untuk LVM.

Langkah-langkah:

# 1. Inisialisasi Physical Volume

pvcreate /dev/sdb /dev/sdc

# 2. Membuat Volume Group

vgcreate vg_data /dev/sdb /dev/sdc

# 3. Membuat Logical Volume dengan ukuran 50GB

lvcreate -L 50G -n lv_storage vg_data

# 4. Format Logical Volume dengan ext4

mkfs.ext4 /dev/vg_data/lv_storage

# 5. Mount ke /mnt/storage

mkdir /mnt/storage

mount /dev/vg_data/lv_storage /mnt/storage

---

# Hasil Implementasi LVM

Pengecekan Physical Volume (PV)

pvdisplay

Output:

--- Physical volume ---

PV Name /dev/sdb

VG Name vg_data

PV Size 100.00 GiB

...

1.

Pengecekan Volume Group (VG)

vgdisplay

Output:

--- Volume group ---

VG Name vg_data

VG Size 200.00 GiB

...

2.

Pengecekan Logical Volume (LV)

lvdisplay

Output:

--- Logical volume ---

LV Name lv_storage

VG Name vg_data

LV Size 50.00 GiB

...

3.

Pengecekan Mount Point

df -h

Output:

Filesystem Size Used Avail Use% Mounted on

/dev/mapper/vg_data-lv_storage 50G 1G 49G 2% /mnt/storage

4.

---

## Kesimpulan

LVM memberikan fleksibilitas dalam manajemen penyimpanan dibandingkan partisi tradisional. Dengan LVM, kita bisa:
✅ Menambah atau mengurangi ukuran volume tanpa kehilangan data.
✅ Menggabungkan beberapa disk menjadi satu kesatuan besar.
✅ Membuat snapshot untuk backup data dengan cepat.

Punya pertanyaan atau mau tes implementasi langsung? 🚀

Oke cuy, gua ajarin LVM dari nol sampe lu paham banget! 🔰🔥

---

# 🔰 Konsep Dasar LVM (Logical Volume Manager)

LVM itu sistem manajemen penyimpanan yang lebih fleksibel dibandingkan partisi biasa. Kalau partisi biasa (MBR/GPT) sifatnya statis, LVM bisa diubah-ubah ukurannya kapan aja tanpa harus format ulang.

## 🔹 Struktur LVM

LVM punya tiga level utama:

1. Physical Volume (PV) → Disk atau partisi yang dijadikan storage untuk LVM.

2. Volume Group (VG) → Gabungan dari satu atau lebih PV yang dijadikan satu kesatuan penyimpanan besar.
3. Logical Volume (LV) → Seperti partisi yang bisa diubah ukurannya tanpa ganggu data.

🎯 Simpelnya: Disk (PV) → Kumpulin jadi satu (VG) → Bagi-bagi buat partisi (LV).

---

# 🔧 Implementasi LVM di Linux

💻 Simulasi ini gua pake dua disk kosong: /dev/sdb & /dev/sdc
Lu bisa cek disk dulu pake:

lsblk

---

## 🛠️ 1. Konversi Disk Jadi PV (Physical Volume)

Kita ubah disk kosong jadi PV dulu.

pvcreate /dev/sdb /dev/sdc

📝 Cek hasilnya:

pvdisplay

---

## 📦 2. Buat Volume Group (VG)

Gabungkan kedua PV ke dalam satu VG (misalnya namanya vg_data).

vgcreate vg_data /dev/sdb /dev/sdc

📝 Cek hasilnya:

vgdisplay

---

## 📁 3. Buat Logical Volume (LV)

Misalnya kita mau bikin partisi lv_storage ukuran 50GB.

lvcreate -L 50G -n lv_storage vg_data

📝 Cek hasilnya:

lvdisplay

---

## 💿 4. Format & Mount LV

Sekarang kita format jadi ext4 dan mount ke /mnt/storage.

mkfs.ext4 /dev/vg_data/lv_storage

mkdir /mnt/storage

mount /dev/vg_data/lv_storage /mnt/storage

📝 Cek apakah sudah ter-mount:

df -h

---

## 📈 Resize LVM (Ubah Ukuran)

LVM itu fleksibel, lu bisa memperbesar atau mengecilkan ukuran LV kapan aja.

### 📌 Cara Tambah Ukuran LV

Misalnya mau tambah 20GB ke lv_storage:

lvextend -L +20G /dev/vg_data/lv_storage

resize2fs /dev/vg_data/lv_storage

---

### 📌 Cara Kecilkan Ukuran LV

⚠️ Hati-hati! Backup dulu data sebelum mengecilkan LV.
Misalnya mau kurangi jadi 30GB:

umount /mnt/storage

e2fsck -f /dev/vg_data/lv_storage

resize2fs /dev/vg_data/lv_storage 30G

lvreduce -L 30G /dev/vg_data/lv_storage

mount /dev/vg_data/lv_storage /mnt/storage

---

## 🎯 Kesimpulan

✅ LVM bikin partisi lebih fleksibel & gampang diatur.
✅ Bisa tambah/kurangi ukuran tanpa format ulang.
✅ Cocok buat server atau storage yang sering berubah.

Kalo ada pertanyaan atau mau coba langsung, gaskeun aja bro! 🚀🔥

Oke cuy, sekarang kita masuk ke LVM tingkat menengah! 🔥
Di level ini, kita bakal bahas fitur snapshots, striping, mirroring, sama moving PV ke disk lain.

---

## 1️⃣ Snapshots (Backup Cepat)

Snapshot itu kaya backup instan dari suatu LV. Ini berguna buat rollback kalau ada kesalahan.

### 🔹 Cara Bikin Snapshot

Misalnya kita punya LV bernama lv_storage, kita bikin snapshot-nya ukuran 10GB:

lvcreate -L 10G -s -n lv_snap /dev/vg_data/lv_storage

📝 Cek hasilnya:

lvdisplay

---

### 🔹 Restore Snapshot ke LV Asli

Misalnya ada kesalahan dan kita mau rollback ke snapshot tadi:

lvconvert --merge /dev/vg_data/lv_snap

Setelah itu reboot sistem biar restore-nya langsung berlaku:

reboot

---

### 🔹 Hapus Snapshot Kalau Udah Gak Dipake

lvremove /dev/vg_data/lv_snap

---

## 2️⃣ Striping (Performa Maksimal)

LVM bisa bagi data ke banyak disk (striping) buat ningkatin performa baca/tulis data. Ini mirip RAID 0.

### 🔹 Buat LV dengan Striping

Misalnya kita punya dua disk (/dev/sdb & /dev/sdc), kita bikin LV yang ter-striping:

lvcreate -L 50G -i 2 -I 64 -n lv_striped vg_data

📝 Keterangan:

- -i 2 → Striping ke 2 disk.
- -I 64 → Ukuran striping 64KB (bisa diubah jadi 128KB atau 256KB).

📌 Striping cocok buat database atau file besar biar baca/tulis lebih cepet.

---

## 3️⃣ Mirroring (Keamanan Data)

Mirroring itu duplikasi data ke lebih dari satu disk, mirip RAID 1.

### 🔹 Buat LV dengan Mirroring

Misalnya kita mau bikin LV mirror di dua disk (/dev/sdb & /dev/sdc):

lvcreate -L 50G -m 1 -n lv_mirror vg_data

📝 Keterangan:

- -m 1 → Mirror 1 copy (jadi total ada 2 salinan).
- Cocok buat storage penting yang butuh redundansi.

## 4️⃣ Migrasi PV ke Disk Lain (Ganti Harddisk)

Misalnya disk /dev/sdb mulai rusak dan kita mau mindahin datanya ke /dev/sdd tanpa downtime.

### 🔹 Tambah Disk Baru ke VG

vgextend vg_data /dev/sdd

### 🔹 Pindahin Data dari Disk Lama ke Disk Baru

pvmove /dev/sdb /dev/sdd

### 🔹 Hapus Disk Lama dari VG

vgreduce vg_data /dev/sdb

pvremove /dev/sdb

💡 Sekarang /dev/sdb udah kosong dan bisa dicabut! 🚀

---

## 🎯 Kesimpulan (LVM Tingkat Menengah)

✅ Snapshot buat backup cepat & rollback data.
✅ Striping buat ningkatin performa baca/tulis data.
✅ Mirroring buat duplikasi data biar lebih aman.
✅ Migrasi PV buat pindahin data ke disk baru tanpa downtime.

Kalo udah paham sini, next kita bisa lanjut ke LVM Advanced (Thin Provisioning, RAID LVM, Auto-resizing, dsb.)! 🔥🔥🔥

Mantap cuy, sekarang kita gas ke LVM tingkat lanjut! 🚀🔥
Di level ini, kita bakal bahas Thin Provisioning, RAID LVM, auto-resizing, dan encryption di LVM.

---

## 1️⃣ Thin Provisioning (Alokasi Dinamis)

Biasanya kalau kita bikin LV, kapasitasnya langsung kepake. Tapi Thin Provisioning bikin LV bisa tumbuh otomatis sesuai kebutuhan, mirip cloud storage.

---

### 🔹 Buat Thin Pool (Storage Dinamis)

Misalnya kita punya VG vg_data, kita buat Thin Pool ukuran 100GB:

lvcreate -L 100G --thinpool thin_pool vg_data

---

### 🔹 Buat Thin Volume (LV yang Bisa Tumbuh)

Kita bikin LV di atas thin_pool tadi:

lvcreate -V 50G --thin -n lv_thin vg_data/thin_pool

💡 Di sini, LV lv_thin cuma bakal pake ruang yang dibutuhkan, walaupun dideklarasikan 50GB.

---

### 🔹 Cek Penggunaan Thin Volume

lvs -a vg_data

Kalau ruang thin pool mau ditambah:

lvextend -L +50G /dev/vg_data/thin_pool

---

## 2️⃣ RAID LVM (RAID Software di LVM)

LVM bisa bikin RAID software buat redundancy dan performa tinggi.
Format RAID yang bisa dipakai di LVM:

- RAID 0 (Striping) → Performa tinggi, tanpa proteksi.
- RAID 1 (Mirroring) → Duplikasi data ke disk lain.
- RAID 5 (Striping + Parity) → Efisien, butuh minimal 3 disk.
- RAID 6 (Double Parity) → Butuh minimal 4 disk.
- RAID 10 (Striping + Mirroring) → Performa + proteksi tinggi.

### 🔹 Buat LVM RAID 5

Misalnya kita punya 3 disk (/dev/sdb, /dev/sdc, /dev/sdd), kita buat RAID 5:

lvcreate -L 100G -i 2 --type raid5 -n lv_raid5 vg_data

💡 Di sini, 1 disk dipakai buat parity (redundansi), jadi total usable storage = (jumlah disk - 1).

---

## 3️⃣ Auto-Resizing (LV yang Bisa Nambah Ukuran Otomatis)

Biasanya LVM butuh resize manual, tapi kita bisa pakai fsadm biar otomatis.

---

### 🔹 Aktifkan Auto-Resize

lvextend -L +10G /dev/vg_data/lv_storage

fsadm resize /dev/vg_data/lv_storage

💡 fsadm akan otomatis resize filesystem tanpa unmount! 🚀

---

## 4️⃣ Enkripsi LVM (Proteksi Data Maksimal)

LVM bisa dienkripsi pake LUKS (Linux Unified Key Setup) buat proteksi data.

---

### 🔹 Enkripsi LV dengan LUKS

Misalnya kita punya LV /dev/vg_data/lv_secure, kita enkripsi:

cryptsetup luksFormat /dev/vg_data/lv_secure

---

### 🔹 Buka Partisi yang Terenkripsi

cryptsetup open /dev/vg_data/lv_secure secure_volume

Sekarang LV ini muncul di /dev/mapper/secure_volume.

---

### 🔹 Format & Mount

mkfs.ext4 /dev/mapper/secure_volume

mount /dev/mapper/secure_volume /mnt/secure

---

## 🎯 Kesimpulan (LVM Tingkat Lanjut)

✅ Thin Provisioning buat storage yang fleksibel.
✅ RAID LVM buat redundancy & performa tinggi.
✅ Auto-Resizing buat LV yang bisa tumbuh otomatis.
✅ Encryption LVM buat proteksi data maksimal.

Sekarang lu udah jadi Master LVM bro! 🔥🔥🔥
Mau lanjut ke materi apa lagi? 😎

Mantap, lu mau dalemin opsi-opsi LVM! 🔥
Oke, gua jelasin satu per satu opsi dari tingkat dasar, menengah, sampe lanjut, biar makin paham.

---

## 📌 Opsi LVM & Fungsinya

### 1️⃣ Opsi Dasar

| Opsi | Fungsi | Contoh Penggunaan |
|---|---|---|
| -L | Menentukan ukuran LV, VG, atau PV dalam MB, GB, atau TB | lvcreate -L 20G -n lv_data vg_main (Buat LV ukuran 20GB) |
| -n | Menentukan nama Logical Volume | lvcreate -L 50G -n lv_backup vg_data (Buat LV bernama lv_backup) |

---

## 2️⃣ Opsi Menengah

| Opsi | Fungsi | Contoh Penggunaan |
|---|---|---|
| -s | Membuat snapshot dari LV | lvcreate -L 10G -s -n lv_snap /dev/vg_data/lv_storage (Buat snapshot lv_snap) |
| --merge | Mengembalikan LV ke kondisi snapshot | lvconvert --merge /dev/vg_data/lv_snap (Restore snapshot) |
| -i | Menentukan jumlah disk untuk striping | lvcreate -L 50G -i 2 -I 64 -n lv_striped vg_data (Striping ke 2 disk) |
| -I | Menentukan ukuran striping dalam KB | lvcreate -L 50G -i 2 -I 128 -n lv_striped vg_data (Gunakan striping 128KB) |
| -m | Membuat mirror volume (RAID 1) | lvcreate -L 50G -m 1 -n lv_mirror vg_data (Buat mirror LV) |

---

## 3️⃣ Opsi Lanjut

| Opsi | Fungsi | Contoh Penggunaan |
|---|---|---|
| --thinpool | Membuat thin pool untuk alokasi dinamis | lvcreate -L 100G --thinpool thin_pool vg_data |
| -V | Membuat volume virtual (Thin Provisioning) | lvcreate -V 50G --thin -n lv_thin vg_data/thin_pool |
| fsadm | Resize filesystem secara otomatis setelah LV diperbesar | fsadm resize /dev/vg_data/lv_storage |
| luksFormat | Meng-enkripsi LV dengan LUKS | cryptsetup luksFormat /dev/vg_data/lv_secure |
| --type raid5 | Membuat RAID 5 di LVM | lvcreate -L 100G -i 2 --type raid5 -n lv_raid5 vg_data |

---

## 🔥 Kesimpulan

- -L & -n → Ukuran & nama LV.
- -s & --merge → Snapshot & restore LV.

- -i & -I → Striping buat performa tinggi.
- -m → Mirroring buat backup otomatis.
- --thinpool & -V → Thin Provisioning buat storage fleksibel.
- fsadm → Auto-resizing tanpa ribet.
- luksFormat → Enkripsi data biar aman.
- --type raid5 → RAID LVM buat kombinasi performa & redundancy.

Sekarang lu udah bisa ngerti semua opsi penting di LVM! 🔥🔥🔥
Mau lanjut ke praktek lebih lanjut atau bahas fitur lain? 😎

Oke cuy, sekarang kita praktek real biar lu makin jago! 🔥

Kita bakal simulasiin beberapa skenario:
1️⃣ Buat LVM dari awal (PV, VG, LV)
2️⃣ Resize LV & Filesystem otomatis
3️⃣ Snapshot & Restore
4️⃣ Thin Provisioning (Storage Dinamis)
5️⃣ RAID LVM (Redundansi)
6️⃣ Enkripsi LV pake LUKS

---

## 1️⃣ Buat LVM dari Nol

🔹 Cek disk kosong yang ada:

```
lsblk
```

🔹 Inisialisasi Physical Volume (PV) di /dev/sdb & /dev/sdc

```
pvcreate /dev/sdb /dev/sdc
```

🔹 Buat Volume Group (VG) vg_storage

```
vgcreate vg_storage /dev/sdb /dev/sdc
```

🔹 Buat Logical Volume (LV) lv_data ukuran 50GB

```
lvcreate -L 50G -n lv_data vg_storage
```

🔹 Format LV dengan ext4 & Mount ke /mnt/data

```
mkfs.ext4 /dev/vg_storage/lv_data
```

mkdir /mnt/data

mount /dev/vg_storage/lv_data /mnt/data

- Cek hasilnya:

df -h /mnt/data

---

## 2 Resize LV & Filesystem Otomatis

- Tambahin 20GB ke LV

lvextend -L +20G /dev/vg_storage/lv_data

fsadm resize /dev/vg_storage/lv_data

- Cek apakah ukuran filesystem bertambah

df -h /mnt/data

---

## 3 Snapshot & Restore

- Buat snapshot lv_backup dari lv_data

lvcreate -L 10G -s -n lv_backup /dev/vg_storage/lv_data

- Restore snapshot kalau ada kesalahan

lvconvert --merge /dev/vg_storage/lv_backup

---

## 4 Thin Provisioning (Storage Dinamis)

- Buat thin pool thin_pool ukuran 100GB

lvcreate -L 100G --thinpool thin_pool vg_storage

- Buat Thin Volume lv_thin ukuran 50GB

lvcreate -V 50G --thin -n lv_thin vg_storage/thin_pool

🔹 Format & Mount LV Thin

```
mkfs.ext4 /dev/vg_storage/lv_thin

mkdir /mnt/thin

mount /dev/vg_storage/lv_thin /mnt/thin
```

---

## 5️⃣ RAID LVM (Redundansi Data)

🔹 Buat RAID 1 (Mirroring) untuk keamanan data

```
lvcreate -L 50G -m 1 -n lv_raid1 vg_storage
```

🔹 Buat RAID 5 (Striping + Parity)

```
lvcreate -L 100G -i 2 --type raid5 -n lv_raid5 vg_storage
```

---

## 6️⃣ Enkripsi LV pake LUKS

🔹 Enkripsi lv_secure

```
cryptsetup luksFormat /dev/vg_storage/lv_secure
```

🔹 Buka LV yang terenkripsi

```
cryptsetup open /dev/vg_storage/lv_secure secure_volume
```

🔹 Format & Mount Encrypted Volume

```
mkfs.ext4 /dev/mapper/secure_volume

mkdir /mnt/secure

mount /dev/mapper/secure_volume /mnt/secure
```

---

## 🔥 Kesimpulan

✅ Udah bisa bikin LVM dari awal
✅ Paham resize otomatis pake fsadm

✅ Bisa snapshot & restore tanpa takut kehilangan data
✅ Ngerti thin provisioning buat storage fleksibel
✅ Paham RAID di LVM buat backup atau performa
✅ Mampu ngamankan data pake LUKS encryption

Udah siap jadi LVM Master bro! 🔥 🔥 🔥
Mau lanjut ke mana lagi? 😎

Mantap cuy! Sekarang kita bahas fitur lain di LVM yang lebih advanced biar makin pro! 🔥 🔥 🔥

---

## 📌 Fitur Lanjutan di LVM

1️⃣ Stripping (RAID 0) → Boost Performa
2️⃣ Mirroring (RAID 1) → Redundansi Data
3️⃣ RAID 5/6 → Gabungan Performa & Redundansi
4️⃣ Cache LV → SSD Cache buat HDD
5️⃣ Moving LV → Pindahin LV tanpa Downtime
6️⃣ Resize Volume Group (VG) → Tambah/Kurangi PV
7️⃣ Auto Mounting → Persistent setelah Reboot
8️⃣ Compression & Deduplication → Hemat Space

---

## 1️⃣ Stripping (RAID 0) → Boost Performa

Gunanya?

- Bagi data ke beberapa disk buat akses lebih cepat
- Cocok buat workload yang butuh speed tinggi

🔹 Buat LV dengan striping 2 disk & stripe size 128KB

lvcreate -L 50G -i 2 -I 128 -n lv_striped vg_storage

🔹 Cek hasilnya:

lvdisplay /dev/vg_storage/lv_striped

🔥 Keuntungan: Speed naik, tapi kalau 1 disk rusak, data hilang semua

---

## 2️⃣ Mirroring (RAID 1) → Redundansi Data

Gunanya?

- Semua data dikloning ke disk lain buat proteksi tinggi

🔹 Buat LV dengan mirror (RAID 1)

lvcreate -L 50G -m 1 -n lv_mirror vg_storage

🔹 Cek status mirroring:

lvs -a -o +devices

🔥 Keuntungan: Data tetap aman kalau 1 disk rusak, tapi butuh 2x kapasitas disk

---

## 3️⃣ RAID 5/6 → Gabungan Performa & Redundansi

- RAID 5: Striping + 1 disk parity → Butuh min. 3 disk
- RAID 6: Striping + 2 disk parity → Butuh min. 4 disk

🔹 Buat RAID 5 (Striping + Parity 1 disk)

lvcreate -L 100G -i 2 --type raid5 -n lv_raid5 vg_storage

🔹 Buat RAID 6 (Striping + Parity 2 disk)

lvcreate -L 100G -i 2 --type raid6 -n lv_raid6 vg_storage

🔥 Keuntungan: Kombinasi performa & proteksi, tapi butuh banyak disk

---

## 4️⃣ Cache LV → SSD Cache buat HDD

Gunanya?

- Gunakan SSD sebagai cache buat HDD biar lebih cepat

🔹 Tambahkan SSD ke VG

vgextend vg_storage /dev/nvme0n1

🔹 Buat cache pool di SSD

lvcreate --type cache-pool -L 20G -n lv_cache vg_storage /dev/nvme0n1

🔹 Pasang cache ke LV utama

lvconvert --type cache --cachevol lv_cache --cachemode writeback vg_storage/lv_data

🔥 Keuntungan: HDD lemot jadi lebih cepat tanpa harus ganti semua ke SSD

---

## 5️⃣ Moving LV → Pindahin LV tanpa Downtime

🔹 Misalnya LV ada di /dev/sdb, mau pindah ke /dev/sdc

pvmove /dev/sdb /dev/sdc

🔥 Keuntungan: Bisa upgrade disk tanpa matiin sistem

---

## 6️⃣ Resize Volume Group (VG) → Tambah/Kurangi PV

🔹 Tambah disk ke VG

vgextend vg_storage /dev/sdd

🔹 Hapus disk dari VG (pastikan kosong dulu)

vgreduce vg_storage /dev/sdb

🔥 Keuntungan: Bisa tambah/hapus disk kapan aja

---

## 7️⃣ Auto Mounting → Persistent setelah Reboot

🔹 Tambahin ke /etc/fstab

echo "/dev/vg_storage/lv_data /mnt/data ext4 defaults 0 0" >> /etc/fstab

🔥 Keuntungan: LV otomatis ter-mount pas boot

---

## 8️⃣ Compression & Deduplication → Hemat Space

Gunanya?

- Kompresi → Nyimpen lebih banyak data di LV
- Deduplication → Hapus data duplikat otomatis

🔹 Gunakan btrfs untuk kompresi otomatis

mkfs.btrfs -d single /dev/vg_storage/lv_data

mount -o compress=zstd /dev/vg_storage/lv_data /mnt/data

🔥 Keuntungan: Hemat space tanpa ribet

---

## 🔥 Kesimpulan

✅ RAID 0 → Boost performa, tapi rawan kehilangan data
✅ RAID 1 → Data aman, tapi boros space
✅ RAID 5/6 → Kombinasi speed + redundancy
✅ SSD Cache → HDD jadi lebih cepat tanpa upgrade penuh
✅ Moving LV → Pindahin disk tanpa downtime
✅ Resize VG → Tambah/hapus disk kapan aja
✅ Auto Mounting → Biar LV tetap aktif setelah reboot
✅ Compression & Deduplication → Hemat space

🔥🔥🔥 Sekarang lu udah jadi LVM PRO! 🔥🔥🔥
Mau lanjut bahas apa lagi? 😎

Yoo, masih lanjut bahas opsi-opsi & jenis-jenis fitur di LVM nih! 🔥
Kita bakal kupas tuntas dari pengertian, cara kerja, sampai contoh implementasi biar makin jago! 😎

---

## 📌 Opsi & Jenis-Jenis LVM (Lengkap)

1️⃣ Standard Logical Volume (LV) → LV biasa
2️⃣ Thin Provisioning → Alokasi dinamis
3️⃣ Snapshot → Backup instan
4️⃣ RAID Logical Volume (RAID 0,1,5,6,10) → Performa & proteksi
5️⃣ Cache Logical Volume → SSD cache buat HDD
6️⃣ Striped Logical Volume → Speed boost

7️⃣ Mirrored Logical Volume → Data aman
8️⃣ Encrypted Logical Volume → LVM + LUKS (security)
9️⃣ Moving Logical Volume → Pindahin LV tanpa downtime
🔟 Resizing Logical Volume → Tambah/Kurangi ukuran LV
1️⃣1️⃣ Thinpool & ThinLV → Volume fleksibel

---

## 1️⃣ Standard Logical Volume (LV) → LV Biasa

🔹 Pengertian:
LV standar adalah partisi virtual dalam LVM yang dibuat dari Volume Group (VG) dan bisa digunakan layaknya partisi biasa.

🔹 Cara Kerja:

1. Buat PV (disk atau partisi)
2. Gabung ke VG
3. Buat LV dari VG

🔹 Contoh Implementasi:

lvcreate -L 20G -n lv_data vg_storage

mkfs.ext4 /dev/vg_storage/lv_data

mount /dev/vg_storage/lv_data /mnt/data

🔹 Hasil:

- Terbentuk LV sebesar 20GB
- Bisa digunakan sebagai penyimpanan biasa

---

## 2️⃣ Thin Provisioning → Alokasi Dinamis

🔹 Pengertian:
Thin Provisioning memungkinkan pembuatan LV dengan kapasitas virtual lebih besar dari fisiknya (misal, buat 500GB tapi cuma pake 50GB dulu).

🔹 Cara Kerja:

1. Buat Thin Pool

2. Buat Thin LV di dalamnya

🔹 Contoh Implementasi:

lvcreate --size 50G --thinpool thin_pool vg_storage

lvcreate --thin -V 500G -n lv_thin vg_storage/thin_pool

mkfs.ext4 /dev/vg_storage/lv_thin

mount /dev/vg_storage/lv_thin /mnt/thin

🔹 Hasil:

- LV bisa bertumbuh otomatis seiring pemakaian

---

## 3️⃣ Snapshot → Backup Instan

🔹 Pengertian:
Snapshot adalah duplikat sementara dari LV yang bisa digunakan untuk recovery tanpa backup penuh.

🔹 Cara Kerja:

1. Snapshot dibuat dari LV asli
2. Snapshot hanya menyimpan perubahan

🔹 Contoh Implementasi:

lvcreate -s -L 10G -n lv_backup /dev/vg_storage/lv_data

🔹 Hasil:

- Bisa revert data ke kondisi sebelum snapshot

---

## 4️⃣ RAID Logical Volume (RAID 0,1,5,6,10)

🔹 Pengertian:
RAID di LVM digunakan untuk meningkatkan performa dan proteksi data dengan beberapa disk.

🔹 Cara Kerja:

1. RAID 0 (Striping) → Speed naik, tapi rawan data hilang
2. RAID 1 (Mirroring) → Redundansi tinggi
3. RAID 5/6 → Kombinasi performa & keamanan
4. RAID 10 → Performa & proteksi tinggi

🔹 Contoh Implementasi (RAID 5):

lvcreate -L 100G -i 2 --type raid5 -n lv_raid5 vg_storage

🔹 Hasil:

- Data lebih aman jika ada disk yang gagal

---

## 5️⃣ Cache Logical Volume → SSD Cache buat HDD

🔹 Pengertian:
Gunakan SSD sebagai cache buat HDD biar lebih cepat.

🔹 Cara Kerja:

1. Tambahkan SSD ke VG
2. Buat cache pool
3. Pasang cache ke LV utama

🔹 Contoh Implementasi:

lvcreate --type cache-pool -L 20G -n lv_cache vg_storage /dev/nvme0n1

lvconvert --type cache --cachevol lv_cache --cachemode writeback vg_storage/lv_data

🔹 Hasil:

- Kecepatan HDD meningkat

---

## 6️⃣ Striped Logical Volume → Speed Boost

🔹 Pengertian:
Data dibagi ke beberapa disk untuk meningkatkan performa I/O.

🔹 Cara Kerja:

1. Gunakan lebih dari 1 disk
2. Set striping saat buat LV

🔹 Contoh Implementasi:

lvcreate -L 50G -i 2 -I 128 -n lv_striped vg_storage

🔹 Hasil:

- Akses data lebih cepat

---

## 7️⃣ Mirrored Logical Volume → Data Aman

🔹 Pengertian:
Data direplikasi ke disk lain untuk redundansi.

🔹 Contoh Implementasi:

lvcreate -L 50G -m 1 -n lv_mirror vg_storage

🔹 Hasil:

- Jika 1 disk rusak, data tetap aman

---

## 8️⃣ Encrypted Logical Volume → LVM + LUKS (Security)

🔹 Pengertian:
LV bisa dienkripsi dengan LUKS untuk keamanan tambahan.

🔹 Contoh Implementasi:

cryptsetup luksFormat /dev/vg_storage/lv_secure

cryptsetup open /dev/vg_storage/lv_secure secure_data

mkfs.ext4 /dev/mapper/secure_data

mount /dev/mapper/secure_data /mnt/secure

🔹 Hasil:

- Data aman dari akses ilegal

---

## 9️⃣ Moving Logical Volume → Pindahin LV tanpa Downtime

🔹 Contoh Implementasi:

```
pvmove /dev/sdb /dev/sdc
```

🔹 Hasil:

- Bisa upgrade disk tanpa shutdown

---

## 🔟 Resizing Logical Volume → Tambah/Kurangi ukuran LV

🔹 Contoh Implementasi (Perbesar LV):

```
lvextend -L +10G /dev/vg_storage/lv_data

resize2fs /dev/vg_storage/lv_data
```

🔹 Hasil:

- LV bertambah 10GB

---

## 1️⃣1️⃣ Thinpool & ThinLV → Volume Fleksibel

🔹 Pengertian:
Thinpool + ThinLV adalah solusi untuk alokasi storage dinamis.

🔹 Contoh Implementasi:

```
lvcreate --thinpool thin_pool -L 50G vg_storage

lvcreate --thin -V 200G -n lv_thin vg_storage/thin_pool
```

🔹 Hasil:

- Bisa buat LV besar tanpa langsung pakai semua storage

---

## 🔥 Kesimpulan

✅ Banyak opsi di LVM buat fleksibilitas, performa, & keamanan
✅ Thin Provisioning & ThinLV → Volume fleksibel
✅ Snapshot → Backup instan
✅ RAID, Striped & Cache LV → Performa meningkat
✅ Encrypted LV → Data lebih aman

🔥🔥🔥 Sekarang lo udah jago LVM Level Dewa! 😎🔥
Mau lanjut bahas apa lagi? 🚀

Mantap lu udah gas LVM sampai level lanjut! 🔥
Sekarang gua bakal bahas opsi-opsi lain yang lebih advance + cara kerjanya biar makin paham.

---

# 🔥 Opsi Tambahan di LVM (Advanced)

Selain yang udah lu sebutin, ada beberapa opsi lagi yang bisa dipake buat optimasi LVM. Gua bakal pecah jadi beberapa kategori biar jelas.

---

## 1️⃣ Opsi Optimasi Storage

| | | |
|---|---|---|
| Opsi | Fungsi | Contoh Penggunaan |
| --alloc contiguous | Memastikan LVM dialokasikan dalam blok yang berdekatan untuk performa tinggi. | lvcreate -L 50G --alloc contiguous -n lv_contig vg_data |
| --zero y | Mengisi sektor dengan nol saat membuat LV baru (biar bersih & aman). | lvcreate -L 10G --zero y -n lv_clean vg_data |
| --wipesignatures y | Menghapus tanda tangan filesystem lama dari LV. | lvcreate -L 20G --wipesignatures y -n lv_safe vg_data |

✅ 🔍 Cara Kerja & Implementasi

- --alloc contiguous cocok buat database atau VM biar baca tulis lebih cepat.
- --zero y dipake saat mau buat LV baru tanpa risiko data lama bocor.

- --wipesignatures y buat bersihin metadata LV sebelumnya supaya nggak bentrok pas format ulang.

🔧 Contoh Hasil
Misal pake lvcreate -L 50G --alloc contiguous -n lv_contig vg_data, LV bakal dibuat dengan blok yang berdekatan, jadi aksesnya lebih cepat.

---

## 2️⃣ Opsi RAID & Redundancy

| Opsi | Fungsi | Contoh Penggunaan |
|---|---|---|
| --type raid1 | Membuat RAID 1 (Mirroring) di LVM. | lvcreate -L 50G --type raid1 -n lv_mirror vg_data |
| --type raid10 | Membuat RAID 10 (Kombinasi striping & mirroring). | lvcreate -L 100G --type raid10 -i 2 -n lv_raid10 vg_data |
| --type raid6 | Membuat RAID 6 (Redundancy lebih tinggi dari RAID 5). | lvcreate -L 100G --type raid6 -i 2 -n lv_raid6 vg_data |

✅ 🔍 Cara Kerja & Implementasi

- raid1 → Buat backup otomatis dengan mirroring, jadi kalau satu disk mati, data masih aman.
- raid10 → Kombinasi RAID 1 + RAID 0, buat performa tinggi + redundancy.
- raid6 → Bisa tahan sampai 2 disk mati sekaligus.

🔧 Contoh Hasil
Misal pake lvcreate -L 100G --type raid6 -i 2 -n lv_raid6 vg_data, data akan disebar di beberapa disk dengan checksum tambahan buat keamanan.

---

## 3️⃣ Opsi Manajemen Volume (Resize & Migration)

| Opsi | Fungsi | Contoh Penggunaan |
|---|---|---|
| lvextend -L +10G | Menambah ukuran LV. | lvextend -L +10G /dev/vg_data/lv_storage |

| | | |
|---|---|---|
| lvreduce -L -10G | Mengurangi ukuran LV. | lvreduce -L -10G /dev/vg_data/lv_storage |
| pvmove | Memindahkan data dari satu PV ke PV lain tanpa downtime. | pvmove /dev/sdb /dev/sdc |

✅ 🔍 Cara Kerja & Implementasi

- lvextend → Kalau butuh lebih banyak storage, bisa perbesar tanpa kehilangan data.
- lvreduce → Hati-hati, kalo salah bisa hilang data! Biasanya dipake buat efisiensi.
- pvmove → Kalau ada disk mau diganti tanpa shutdown server, bisa mindahin datanya dulu ke disk lain.

🔧 Contoh Hasil
Misal pake pvmove /dev/sdb /dev/sdc, semua data dari /dev/sdb bakal dipindah ke /dev/sdc tanpa matiin sistem.

---

## 4️⃣ Opsi Keamanan & Enkripsi

| Opsi | Fungsi | Contoh Penggunaan |
|---|---|---|
| cryptsetup luksFormat | Meng-enkripsi LV dengan LUKS. | cryptsetup luksFormat /dev/vg_data/lv_secure |
| cryptsetup open | Membuka enkripsi LVM. | cryptsetup open /dev/vg_data/lv_secure lv_decrypt |

✅ 🔍 Cara Kerja & Implementasi

- luksFormat buat proteksi data biar nggak bisa diakses sembarangan.
- cryptsetup open buat buka enkripsi pas mau akses datanya.

🔧 Contoh Hasil
Misal pake cryptsetup luksFormat /dev/vg_data/lv_secure, data di LV itu bakal terenkripsi & butuh password buat diakses.

---

## 🔥 Kesimpulan

LVM tuh fleksibel banget! Lo bisa:
✅ Optimasi storage dengan --alloc contiguous, --zero y, dll.
✅ Buat RAID LVM pake --type raid1/raid10/raid6.
✅ Resize & migrasi LV dengan lvextend, lvreduce, pvmove.
✅ Amankan data pake cryptsetup luksFormat buat enkripsi.

---

Mau lanjut bahas yang mana lagi? Mungkin langsung ke simulasi prakteknya? 😎

🔥 Let's go! Simulasi Praktik LVM dari Nol! 🔥

Kita bakal bikin skenario lengkap mulai dari inisialisasi LVM, bikin Volume Group, Logical Volume, resize, snapshot, RAID, sampai enkripsi.

---

## 🎯 Skenario LVM yang Kita Buat

1️⃣ Persiapan & Setup: Tambahin disk, cek disk, dan inisialisasi PV.
2️⃣ Bikin Volume Group (VG) & Logical Volume (LV).
3️⃣ Resize LV (Perbesar & Perkecil ukuran).
4️⃣ Snapshot & Restore Data dari Snapshot.
5️⃣ Implementasi RAID di LVM.
6️⃣ Pindah data antar disk tanpa downtime.
7️⃣ Enkripsi LV dengan LUKS.

---

## 🛠 1️⃣ Persiapan & Setup LVM

🔥 Cek daftar disk yang belum dipakai:

lsblk

🔥 Tambahin disk baru ke sistem (Misal: /dev/sdb dan /dev/sdc).
Kalau pakai VirtualBox, bisa tambahin disk via Settings > Storage > Add Disk.

🔥 Inisialisasi disk buat LVM:

pvcreate /dev/sdb /dev/sdc

✅ Hasilnya:
Cek status PV:

pvs

Output-nya bakal ada /dev/sdb dan /dev/sdc yang udah jadi PV.

---

## 📌 2️⃣ Membuat Volume Group (VG) & Logical Volume (LV)

🔥 Buat Volume Group (VG) baru dari PV tadi:

vgcreate vg_data /dev/sdb /dev/sdc

🔥 Cek status VG:

vgs

🔥 Buat Logical Volume (LV) ukuran 20GB:

lvcreate -L 20G -n lv_storage vg_data

🔥 Cek status LV:

lvs

🔥 Format LV dengan EXT4 dan mount ke /mnt/data:

mkfs.ext4 /dev/vg_data/lv_storage

mkdir /mnt/data

mount /dev/vg_data/lv_storage /mnt/data

🔥 Cek apakah berhasil:

df -h | grep /mnt/data

🔥 Agar auto-mount saat reboot, edit /etc/fstab:

echo "/dev/vg_data/lv_storage /mnt/data ext4 defaults 0 2" >> /etc/fstab

---

## 🔧 3️⃣ Resize LV (Perbesar & Perkecil ukuran)

🔥 Tambah 10GB ke LV:

lvextend -L +10G /dev/vg_data/lv_storage

resize2fs /dev/vg_data/lv_storage

🔥 Kurangi ukuran LV (⚠ Hati-hati, bisa kehilangan data!):

umount /mnt/data

lvreduce -L -5G /dev/vg_data/lv_storage

resize2fs /dev/vg_data/lv_storage

mount /dev/vg_data/lv_storage /mnt/data

---

## 📸 4️⃣ Snapshot & Restore Data dari Snapshot

🔥 Buat snapshot dari LV utama:

lvcreate -L 5G -s -n lv_snap /dev/vg_data/lv_storage

🔥 Restore dari snapshot (kalau ada kesalahan di LV utama):

lvconvert --merge /dev/vg_data/lv_snap

---

## ⚡ 5️⃣ Implementasi RAID di LVM

🔥 Buat LV dengan RAID 1 (Mirroring):

lvcreate -L 20G --type raid1 -m 1 -n lv_mirror vg_data

🔥 Buat LV dengan RAID 5 (Striping + Parity):

lvcreate -L 50G --type raid5 -i 2 -n lv_raid5 vg_data

🔥 Cek status RAID di LV:

lvs -a -o +devices

---

## 🔄 6️⃣ Migrasi Data Antar Disk Tanpa Downtime

🔥 Misal lo mau pindahin dari /dev/sdb ke /dev/sdd:

pvmove /dev/sdb /dev/sdd

🔥 Hapus PV yang udah gak kepake:

vgreduce vg_data /dev/sdb

pvremove /dev/sdb

---

## 🔐 7️⃣ Enkripsi LV dengan LUKS

🔥 Enkripsi LV biar data lebih aman:

cryptsetup luksFormat /dev/vg_data/lv_secure

🔥 Buka & Gunakan LV yang terenkripsi:

cryptsetup open /dev/vg_data/lv_secure lv_decrypt

mkfs.ext4 /dev/mapper/lv_decrypt

mount /dev/mapper/lv_decrypt /mnt/secure

🔥 Kalau mau unmount & kunci LV:

umount /mnt/secure

cryptsetup close lv_decrypt

---

## 🔥 Kesimpulan & Hasil Simulasi

✓ Buat Volume Group & Logical Volume → VG & LV berhasil dibuat.
✓ Resize LV → Tambah/kurangi ukuran LV tanpa hapus data.
✓ Snapshot & Restore → Bisa rollback data dengan cepat.
✓ RAID di LVM → Buat RAID 1 & RAID 5 untuk performa & keamanan.
✓ Migrasi Data → Pindahin data antar disk tanpa downtime.
✓ Enkripsi LV → Amankan data dengan LUKS.

🔥 Done! Sekarang lo udah pro LVM! 🔥

Mau lanjut tes langsung di VM atau ada yang mau dibahas lagi? 😎

Siap beb! Kita bakal ngulik lebih dalem 3 direktori ini:

---

## 🔧 /dev – Device Files (Virtual Filesystem)

### 🧠 Penjelasan:

Direktori /dev adalah virtual filesystem yang digunakan untuk mewakili device fisik atau virtual sebagai file. Ini dikendalikan oleh kernel dan udev (device manager user space).

### 🔹 Contoh isi & fungsinya:

| File | Fungsi |
|---|---|
| /dev/sda | Seluruh disk SATA pertama (biasanya hardisk/SSD utama) |
| /dev/sda1 | Partisi pertama dari /dev/sda |
| /dev/loop0 s/d /dev/loopX | Virtual disk file (loopback, misalnya file .iso) |
| /dev/null | Blackhole Linux. Segala input ke sini akan dibuang (misal: command > /dev/null) |
| /dev/zero | Menghasilkan byte 0, biasa untuk bikin file dummy (dd if=/dev/zero of=file) |
| /dev/random & /dev/urandom | Generator angka random (penting buat crypto/security) |
| /dev/tty | Terminal yang sedang aktif |
| /dev/pts/* | Pseudo terminal (misal saat SSH-an atau pakai terminal emulator) |
| /dev/cdrom, /dev/sr0 | Optical disk drive |

### 🔐 Keamanan & Pemanfaatan:

- Bisa digunakan attacker untuk akses disk langsung via dd.
- Bisa dibuat rule hardening: block akses langsung ke device, terutama /dev/mem, /dev/kmem.

---

## 💿 /boot – Boot Loader dan Kernel

### 🧠 Penjelasan:

Berisi file penting buat booting sistem Linux. Ini diproses oleh bootloader seperti GRUB, sebelum kernel Linux dijalankan.

### 🔹 Contoh isi:

| | |
|---|---|
| File/Folder | Keterangan |
| /boot/vmlinuz-* | File kernel Linux yang akan diload |
| /boot/initrd.img-* | Initial RAM Disk (buat load driver awal saat booting) |
| /boot/grub/ | File konfigurasi bootloader GRUB |
| /boot/System.map-* | Symbol table dari kernel (buat debugging kernel) |

### 🔐 Potensi Risiko:

- Jika file di sini corrupt → system gagal booting
- File boot bisa dimodifikasi attacker untuk tanamkan malware bootkit
- Perlu permission strict & bisa dienkripsi via LUKS + GRUB password

---

## 📂 /mnt – Mount Point Manual

### 🧠 Penjelasan:

Folder ini disediakan untuk admin buat mounting manual device. Tidak digunakan otomatis oleh system seperti /media.

### 🔹 Penggunaan Umum:

# Mount manual flashdisk/hardisk

mount /dev/sdb1 /mnt

# Mount ISO

mount -o loop kali.iso /mnt

### 🔐 Tips Keamanan:

- Untrusted device jangan sembarangan dimount.
- Gunakan opsi nosuid, noexec, nodev saat mount:

mount -o nosuid,noexec,nodev /dev/sdb1 /mnt

- Hapus mount setelah dipakai → umount /mnt

---

Kalau lu mau, kita bisa lanjut ke:

- 💣 Eksploitasi via /dev (ex: raw disk overwrite)
- 🔐 Hardening /boot + GRUB password
- 🧰 Simulasi forensic device mount manual via /mnt

Mau dikerjain yang mana dulu beb? 😘

Oke beb, kita bahas konsep mount dari nol dulu biar lu paham sampe ke akar-akarnya ya 😘
💻🌳

---

## 🧠 Apa Itu Mount di Linux?

Di Linux, mounting artinya “menyambungkan” perangkat penyimpanan (kayak flashdisk, harddisk, ISO, dll) ke filesystem Linux, biar kita bisa akses isinya.

Linux gak langsung ngenalin disk dengan huruf seperti di Windows (C:, D:), tapi pake sistem mount. Jadi setiap storage harus dimount dulu ke folder, baru bisa diakses.

---

## 📌 Contoh Analogi Simpel

Bayangin /mnt itu meja kerja kosong, dan /dev/sdb1 itu flashdisk. Kalau lu mau baca isi flashdisk, harus taruh dulu di meja kerja:

mount /dev/sdb1 /mnt

📁 Maka isi flashdisk bisa diakses lewat /mnt.

Kalau lu udah selesai pakai, lu "angkat lagi flashdisk dari meja":

umount /mnt

## 🔧 Langkah Praktis Mount Manual

### ✅ Cek device dulu:

lsblk

Misalnya muncul:

sdb      8:16   1  7.5G  0 disk

└─sdb1   8:17   1  7.5G  0 part

Artinya ada flashdisk di /dev/sdb1.

### ✅ Mount ke /mnt:

sudo mount /dev/sdb1 /mnt

### ✅ Akses isi:

cd /mnt

ls

### ✅ Kalau udah selesai:

sudo umount /mnt

---

## 🔐 Opsi Keamanan Saat Mount

Kalau kamu mount device dari luar (misalnya flashdisk pinjeman, ISO gak jelas), tambahin opsi:

sudo mount -o nosuid,noexec,nodev /dev/sdb1 /mnt

| | |
|---|---|
| Opsi | Fungsi |
| nosuid | Nonaktifin file dengan SUID/SGID (cegah privilege escalation) |
| noexec | Nonaktifin file executable (gak bisa ./file) |
| nodev | Abaikan file device khusus di flashdisk (biar gak bisa spoof /dev/*) |

---

## 📀 Mount File ISO (Loop Device)

Kalau punya file .iso, lu bisa treat dia kayak disk virtual:

sudo mount -o loop kali-linux.iso /mnt

Lalu isi ISO bisa diakses di /mnt.

---

## 🧹 Kenapa Harus Unmount?

Karena kalau nggak di-unmount, bisa:

- ❌ Data belum disimpan sempurna
- 💣 Terjadi corruption
- 💀 Nggak bisa cabut device dengan aman

Jadi selalu umount sebelum cabut flashdisk ya sayang 😘

---

Kalau mau lanjut simulasi:

✅ Mount berbagai device
✅ Buat file .img sendiri lalu mount
✅ Coba mount dengan permission berbeda

Gue siap pandu lu beb! Mau lanjut yang mana dulu?

Wihh pertanyaan bagus banget beb 😘💻 — jawabannya tergantung distro dan layanan sistemnya!

---

## 🔍 Jawaban Singkat:

### 🔹 Secara default (Linux base/core tanpa GUI):

Linux nggak otomatis mount flashdisk/hardisk eksternal.
Lu harus mount manual pake:

sudo mount /dev/sdb1 /mnt

---

## 🔹 Tapi kalau kamu pakai GUI Desktop Environment (seperti GNOME, KDE, XFCE):

Linux bisa auto-detect dan auto-mount flashdisk/harddisk eksternal
karena dibantu layanan seperti:

- udisks2
- gvfs
- udisksctl
- autofs

Contohnya kalau kamu pakai Ubuntu Desktop, PopOS, Fedora Workstation:
🟢 Colok flashdisk → muncul popup → langsung bisa akses di /media/namaflashdisk.

---

## 💡 Bedanya Auto-Mount vs Manual Mount

| | | |
|---|---|---|
| Fitur | Auto-Mount (GUI) | Manual Mount (CLI/server) |
| Butuh perintah | ❌ Tidak | ✅ Iya (misalnya mount /dev/sdb1 /mnt) |
| Cocok untuk | Pengguna biasa, desktop | Sysadmin, server, terminal-only distro |
| Aman gak? | Kadang rawan jika gak dibatasi | Lebih aman karena butuh izin root |
| Mount path | Biasanya /media/username/label | Biasanya /mnt atau /media |

---

## 🛑 Catatan Penting

- Kalau kamu pakai Linux server kayak CentOS minimal, Debian minimal, Kali CLI, biasanya gak auto-mount sama sekali.
- Buat mount otomatis di server CLI, kamu bisa pakai:
- autofs
- fstab entry
- udev rules custom

Kalau kamu mau:
✅ Simulasi colok flashdisk → mount manual
✅ Set auto-mount di server CLI
✅ Simulasi detect & mount .iso dari file

Bilang aja, nanti kita bahas satu-satu praktiknya sayang 💞

Nah ini dia beb, gua bakal jelasin kapan masing-masing file di /dev/ itu dipakai secara nyata dalam sistem Linux — baik otomatis oleh sistem, maupun pas kita eksploitasi atau administrasi manual. Langsung aja kita gas! 🚀

---

## 📦 /dev Files — Kapan Dipakai & Contoh Kasusnya

| File | Digunakan Saat | Contoh Kasus Nyata |
|---|---|---|
| /dev/sda | Saat lo akses seluruh disk fisik, biasanya pas setup partisi, backup, atau forensic | fdisk /dev/sda, dd if=/dev/sda of=backup.img |
| /dev/sda1 | Saat lo mau mount partisi spesifik | mount /dev/sda1 /mnt buat buka partisi |
| /dev/loop0 s/d /dev/loopX | Saat mount file image (ISO, IMG) seolah-olah disk | mount -o loop kali.iso /mnt |
| /dev/null | Buat buang output/error yang gak dibutuhin | ls / > /dev/null, atau disable log script: script.sh > /dev/null 2>&1 |
| /dev/zero | Bikin file dummy atau format disk dengan nol | dd if=/dev/zero of=1GBfile bs=1M count=1024 |
| /dev/random & /dev/urandom | Dipakai pas generate key encryption, token, GPG, SSL | openssl rand -hex 16, gpg --gen-key |
| /dev/tty | Saat script perlu interaksi langsung ke terminal aktif | echo "Password:" > /dev/tty biar muncul walau script jalan background |
| /dev/pts/* | Saat SSH masuk atau pakai terminal emulator, dia mapping ke pseudo-terminal | who atau tty waktu SSH: /dev/pts/0 |
| /dev/cdrom, /dev/sr0 | Saat lo mau baca CD/DVD, mount ISO dari CD | mount /dev/cdrom /mnt atau dd if=/dev/sr0 of=image.iso |

## 💥 BONUS: Use-case yang sering dipakai hacker/sysadmin

/dev/null → Buat nyembunyiin jejak script atau brute-force agar gak spam log:

hydra -l admin -P passlist.txt ssh://10.10.10.1 > /dev/null 2>&1

- 

/dev/zero → Buat wipe disk biar gak bisa di-recover datanya:

dd if=/dev/zero of=/dev/sda bs=1M

- 

/dev/loopX → Mount file .img hasil forensic CTF:

mount -o loop disk.img /mnt

- 
- /dev/tty → Dipakai tools yang minta input dari user walau di background shell.

---

Kalau lo mau kita praktek langsung:

- Simulasi forensic pake dd + loop device
- Nyoba overwrite disk pake /dev/zero
- Generate key crypto dari /dev/urandom

Tinggal bilang aja ya sayang 🔐 🖤
Mau lanjut ke eksploitasi, forensic, atau simulasi admin?

Yes betul banget beb! 💿 💻 🔥
Kalau lo mau masukin file ISO (misalnya RedHat ISO) ke dalam VM yang lo jalanin — misalnya file .ova yang isinya RedHat — lo bisa banget pakai mount atau akses via /dev/sr0 atau /dev/cdrom tergantung kasusnya.

Tapi nih sayang, kita perlu bedain dulu 2 skenario ya, biar gak keliru:

---

## 💡 Skenario 1: Kamu Pakai ISO ke VM Sebagai Installer

Kalau lo mau boot dari ISO di VM (misalnya install RedHat di VM yang baru atau nambah repo dari ISO):

### ✅ Caranya:

1. Masuk ke VirtualBox / VMware settings
2. Di bagian Storage / CD/DVD Drive, attach ISO kamu (misal rhel.iso)
3. Pas VM jalan, file ISO itu bakal muncul di VM sebagai /dev/cdrom atau /dev/sr0

Di dalam VM RedHat-nya:

mount /dev/cdrom /mnt

# atau

mount /dev/sr0 /mnt

4.
5. Sekarang isi ISO bisa dibaca di /mnt

📌 Contoh Gunanya:

- Akses repo dari DVD install
- Copy file RPM dari ISO
- Mount ISO buat pakai AppStream/installer RedHat

---

## 💡 Skenario 2: Lo Udah Punya VM (misalnya RedHat.ova), dan Mau Tambah File ISO ke Dalamnya

Kalo lo udah punya .ova (VM yang udah dikemas lengkap), terus lo ingin inject file ISO atau mount-nya ke dalam VM, tinggal attach ISO ke VM yang udah ada:

### ✅ Langkahnya:

1. Import dulu .ova ke VirtualBox
2. Buka Settings > Storage
3. Klik ikon CD > Pilih file ISO
4. Jalankan VM

Di dalam VM:

lsblk  # buat lihat device ISO (biasanya sr0)

mount /dev/sr0 /mnt

ls /mnt  # isi file ISO-nya

5.

---

## 🎯 Perlu Diingat:

- /dev/sr0 atau /dev/cdrom muncul kalau dan hanya kalau ISO di-attach ke VM
- ISO bisa dianggap "CD-ROM virtual"
- Kalau lo gak lihat /dev/sr0, kemungkinan ISO belum di-attach dengan benar

---

Mau gua tunjukin simulasi real mount ISO dalam VM step-by-step (pakai screenshot atau text walkthrough)?
Atau mau langsung lanjut ke repo offline RedHat dari ISO? 😏

Pertanyaan lu cakeeep banget beb 😘🔍 — ini sering bikin orang bingung, padahal penting banget di dunia sysadmin & cyber security.

---

## 🔥 Pertanyaan:

“Kan udah ada .ova (Linux-nya udah ke-install), ngapain attach ISO lagi?”

---

## 🧠 Jawaban Singkat:

🔸 Karena ISO itu gak cuma buat install OS doang, tapi juga bisa:

1. 🛟 Jadi repo lokal (RedHat, CentOS, dkk)
2. 🧩 Bawa package .rpm tambahan
3. 🔐 Digunakan buat audit atau patching offline
4. ⚙️ Digunakan buat recovery, repair, atau debug

5. 🔁 Custom media, seperti ISO tools forensic, ISO update image, dsb

---

## 🧭 Analoginya:

Bayangin .ova = PC udah jadi.
Tapi lu butuh CD installer Windows/MS Office buat nambahin fitur, atau CD driver bawaan buat akses jaringan atau media, nah ISO = CD/DVD itu.

---

## ✅ Contoh Kasus Penggunaan ISO di VM:

| | |
|---|---|
| Tujuan | Kenapa Butuh ISO |
| 🔗 Install RPM dari DVD RedHat | RedHat punya repo "AppStream" & "BaseOS" yang kadang cuma ada di ISO |
| 💻 VM tanpa internet (airgap) | Butuh ISO supaya bisa install package tanpa yum update via online |
| 🛠️ Restore system | ISO RedHat bisa boot ke rescue mode |
| 🔁 Nambah fitur/fix | Misal yum install cockpit tapi repo online gak bisa — pake ISO |
| 🔐 Forensik | ISO bisa berupa tools (Kali ISO, GRML, CAINE) yang dimount ke VM untuk analisis |
| 📦 Custom script di ISO | Tim dev bisa simpan script, config, rpm khusus ke dalam ISO lalu kasih ke user |
| 🧱 Hardening LKS | Soal LKS sering minta: “Attach ISO, install package dari repo DVD” |

---

## 💡 Contoh Real Dunia Nyata:

mount /dev/sr0 /mnt

cd /mnt

ls

# Muncul: AppStream, BaseOS, .treeinfo

# Tambah repo dari ISO:

sudo dnf config-manager --add-repo=file:///mnt/BaseOS

# Install rpm dari repo DVD

sudo dnf install vim cockpit chrony

---

Kalau mau gua buatin step-by-step cara attach ISO dan bikin repo offline dari ISO (di VM RedHat/CentOS), tinggal bilang:

“Gaskin praktik repo offline RedHat beb 💿🔥”

Atau kalo mau versi LKS-nya:

“Gue mau versi soal LKS-nya ya beb, buat simulasi langsung 💡”

Wajar banget beb! 💡 Emang mount itu buat pemula keliatan kayak ilmu hitam Linux — padahal sebenernya dia itu konsep dasar tapi super penting, apalagi kalo lu mainin file, disk, ISO, sampe forensic. Gua bantu uraiin sampe ke akarnya ya 😤🌳

---

## 🧠 Apa Itu mount?

mount = ngasih jalan biar isi dari device (disk, ISO, USB, dll) bisa dibaca di Linux.

➡️ Device hardware itu gak bisa langsung dibuka. Harus dipasang dulu ke folder, baru bisa diakses.
Ini disebut "mounting".

---

## 📦 Analogi Gampang:

Bayangin flashdisk = kardus isi barang.
Tapi kardus ini harus dibuka di suatu tempat (misal di meja /mnt) biar bisa lo ambil isinya.

Nah, perintah:

mount /dev/sdb1 /mnt

= "Tolong buka kardus /dev/sdb1 dan tuangin isinya ke meja /mnt"

---

## 🔍 Struktur Perintah mount

mount [opsi] [device] [mount-point]

| | | |
|---|---|---|
| Bagian | Contoh | Artinya |
| [device] | /dev/sdb1 | Nama disk/partisi yang mau dibuka |
| [mount-point] | /mnt | Folder tempat isi device ditampilkan |
| [opsi] | -o ro,nosuid | Opsi tambahan (baca-only, noexec, keamanan, dll) |

---

## 🔧 Contoh Sederhana: Mount USB

sudo mount /dev/sdb1 /mnt

cd /mnt

ls

# Isinya muncul

Kalau udah selesai:

sudo umount /mnt

---

## 📀 Contoh Mount ISO

sudo mount -o loop redhat.iso /mnt

cd /mnt

ls

# Muncul isi ISO (AppStream, BaseOS, dll)

## 🔐 Opsi Keamanan Saat Mount:

sudo mount -o ro,nosuid,noexec,nodev /dev/sdb1 /mnt

| | |
|---|---|
| Opsi | Arti |
| ro | Read Only, biar gak bisa ditulis |
| nosuid | Matikan file SUID (biar gak bisa escalate root) |
| noexec | Gak bisa eksekusi file dalam disk itu |
| nodev | Jangan perlakukan device file (misalnya /dev/xyz) di disk itu |

---

## 🎯 Kapan Lu Gunain mount?

| | |
|---|---|
| Kasus | Perintah |
| Mount flashdisk | mount /dev/sdb1 /mnt |
| Mount ISO | mount -o loop file.iso /mnt |
| Mount partisi khusus | mount /dev/sda3 /mnt/data |
| Mount CD/DVD di VM | mount /dev/sr0 /mnt |
| Mount disk forensik .img | mount -o loop evidence.img /mnt |
| Mount recovery rescue.img | mount -o ro,rescue.img /mnt |

---

Kalau lu udah paham ini, lu udah megang kunci akses file & storage Linux 🔐🔥
Mau gua buatin simulasi soal atau praktek langsung repo dari ISO habis ini?

Ketik aja:

“Gua siap praktik repo offline RedHat dari ISO beb 🛟”
Atau:
“Simulasi soal LKS-nya dong beb 🎯”

Siap, beb! 🛟🔥 Kita langsung praktik bikin repo offline dari ISO Red Hat di VM lu. Ini penting banget buat install paket tanpa internet, apalagi buat LKS atau server airgap.

## 🛠️ Langkah 1: Mount ISO ke VM

### 🧩 Opsi A: Mount ISO File Langsung

sudo mkdir -p /mnt/disc

sudo mount -o loop rhel-8.6-x86_64-dvd.iso /mnt/disc

📌 Ganti rhel-8.6-x86_64-dvd.iso dengan nama file ISO kamu.

### 🧩 Opsi B: Mount dari CD/DVD Virtual (misalnya /dev/sr0)

sudo mkdir -p /mnt/disc

sudo mount /dev/sr0 /mnt/disc

📌 Pastikan ISO sudah di-attach ke VM via VirtualBox/VMware.

---

## 🛠️ Langkah 2: Salin dan Edit File Repo

sudo cp /mnt/disc/media.repo /etc/yum.repos.d/rhel8.repo

sudo chmod 644 /etc/yum.repos.d/rhel8.repo

Edit file repo:

sudo nano /etc/yum.repos.d/rhel8.repo

Ubah isinya menjadi:

[dvd-BaseOS]

name=DVD for RHEL - BaseOS

baseurl=file:///mnt/disc/BaseOS

enabled=1

gpgcheck=1

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

[dvd-AppStream]

name=DVD for RHEL - AppStream

baseurl=file:///mnt/disc/AppStream

enabled=1

gpgcheck=1

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

📌 Pastikan file RPM-GPG-KEY-redhat-release ada di /etc/pki/rpm-gpg/.

---

## 🛠️ Langkah 3: Bersihkan Cache dan Cek Repo

sudo dnf clean all

sudo dnf repolist enabled

📌 Harus muncul dvd-BaseOS dan dvd-AppStream.

---

## 🛠️ Langkah 4: Install Paket dari Repo Offline

sudo dnf --disablerepo="*" --enablerepo="dvd-AppStream" install vim

📌 Ganti vim dengan nama paket yang ingin kamu install.

---

## ✅ Tips Tambahan

- ISO harus tetap ter-mount selama penggunaan repo.
- Jangan reboot atau unmount ISO saat masih diperlukan.
- Gunakan opsi --disablerepo="*" untuk memastikan hanya repo offline yang digunakan.

---

## 📺 Video Tutorial

Butuh panduan visual? Cek video ini:

[Local YUM REPO on CentOS 7 / RHEL 7 using DVD ISO]

---

Kalau kamu mau lanjut ke simulasi soal LKS atau praktek install paket tertentu, tinggal bilang aja, beb! 😎🔥

Mantap bro lu udah mulai ngulik perintah lsblk 🪛 Ini salah satu perintah terbaik buat ngebaca struktur penyimpanan di Linux. Yuk gua jelasin sampai akar-akarnya 👇

---

## 📦 Cara Membaca Output lsblk

Berikut adalah hasil output lu:

```
NAME                   MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINTS
sda                      8:0    0   64G  0 disk
├─sda1                   8:1    0    1G  0 part  /boot
└─sda2                   8:2    0   63G  0 part
  └─luks-90472a40-...  253:0    0   63G  0 crypt
    ├─cs-root          253:1    0 40.9G  0 lvm   /
    ├─cs-swap          253:2    0    2G  0 lvm   [SWAP]
    └─cs-home          253:3    0   20G  0 lvm   /home
sr0                     11:0    1 58.5M  0 rom   /run/media/centos/VBox_GAs_7.1.8
```

---

## 🧠 Penjelasan Setiap Kolom:

| | |
|---|---|
| Kolom | Arti |
| NAME | Nama device/partisi |
| MAJ:MIN | Major & minor device number (untuk kernel/device mapper) |
| RM | Removable? 1 = bisa dicabut (contoh flashdisk), 0 = internal |

| | |
|---|---|
| SIZE | Ukuran total partisi atau device |
| RO | Read-Only? 1 = cuma bisa dibaca, 0 = bisa dibaca-tulis |
| TYPE | Tipe: disk, part, lvm, crypt, rom, dll |
| MOUNTPOINTS | Lokasi mount-nya di filesystem (misal /, /home, /boot, dll) |

---

## 🔍 Breakdown per Bagian

### 🧊 sda → Disk utama (64 GB)

- Ini adalah hard disk atau SSD utama kamu.

### 📦 sda1 → Partisi 1 (1 GB)

- Tipe: part
- Digunakan untuk: /boot — tempat kernel & grub disimpan

### 📦 sda2 → Partisi 2 (63 GB)

- Tipe: part
- Di-encrypt menggunakan LUKS → jadi muncul sebagai:

---

### 🔐 luks-XXXX → Encrypted disk (63 GB)

- Tipe: crypt (LUKS encryption)
- Di dalamnya ada Volume Group (VG) yang berisi 3 Logical Volume (LV):

---

### 📁 cs-root → 40.9 GB → Mount: /

- Sistem utama Linux lu di sini

### 📁 cs-swap → 2 GB → SWAP (Virtual RAM)

- Untuk cadangan memori RAM

## 📁 cs-home → 20 GB → Mount: /home

- Buat data user (file pribadi, config user, dll)

---

## 💿 sr0 → CD/DVD-ROM (ISO VBox Guest Additions)

- Tipe: rom
- Ter-mount otomatis di /run/media/centos/VBox_GAs_7.1.8

---

## 📚 Tipe-Tipe TYPE Penting

| | |
|---|---|
| Tipe | Keterangan |
| disk | Fisik disk utuh |
| part | Partisi pada disk |
| lvm | Logical Volume dari LVM |
| crypt | Encrypted volume (contoh: LUKS) |
| rom | Optical media: CD/DVD |

---

## 📌 Ringkasan Visual

sda (64 GB)

├─ sda1 (1 GB) → /boot

└─ sda2 (63 GB, encrypted)

  └─ luks-xxxxxx

    ├─ cs-root (40.9 GB) → /

    ├─ cs-home (20 GB) → /home

    └─ cs-swap (2 GB) → [SWAP]

sr0 (58.5 MB ISO VBox) → /run/media/centos/VBox_GAs_7.1.8

---

Kalau lo mau lanjut belajar:

- Gimana cara resize partisi
- Bikin partisi baru dari kosong
- Pakai LVM manual
- Cek space pakai df -h, du, dll

Tinggal bilang:

“Lanjut ngoprek disk management manual beb 🔥”

Atau lo mau lanjut ke filesystem lainnya?

Nice banget pertanyaannya beb 🔐🔥 lo udah masuk ke topik LUKS (Linux Unified Key Setup) dan Disk Encryption — ini salah satu praktik keamanan penting dalam dunia Cyber Security & Forensik Digital!

---

## 💡 1. Apa Itu LUKS Encryption?

LUKS = Linux Unified Key Setup

LUKS adalah sistem enkripsi standar di Linux yang mengunci partisi disk, biar data gak bisa diakses sembarangan walau harddisk-nya dicopot dan dibaca di sistem lain.

Kalau lo ngatur passphrase saat install, maka partisi lu akan:

🧱 Ter-enkripsi full → isinya bakal nggak bisa dibaca kalau gak punya kunci/passphrase
🔐 Aman banget buat data pribadi, file sensitif, rahasia kantor, dsb
🛡️ Anti forensik! Cocok buat anak CTF, hacker whitehat, dan tim red team

---

## 🧠 2. Kenapa Gunakan Passphrase (Full Disk Encryption)?

| | |
|---|---|
| Tujuan | Penjelasan |

| | |
|---|---|
| 🛡️ Keamanan Fisik | Kalau laptop lu dicuri, orang gak bisa baca isi disk-nya tanpa passphrase |
| 💉 Anti-Forensik | Data di partisi /home, /var, /tmp, bahkan / akan terenkripsi |
| 🔐 Confidential | Buat data sensitif: file kantor, rahasia, source code, dsb |
| ☠️ Risiko Tanpa Enkripsi | Orang tinggal colok SSD ke PC lain & bisa baca semua data |

---

## 💭 3. Gimana Kalau Pas Install Gak Diatur?

Kalau waktu install lo skip bagian set passphrase:

🚫 Maka CentOS gak akan pakai enkripsi sama sekali
📂 Semua data di /, /home, dsb, akan disimpan dalam bentuk terbuka
💥 Data bisa diekstrak pakai live USB atau OS lain

---

## 🔧 4. Bisa Gak Enkripsi Setelah CentOS Sudah Terlanjur Terinstall?

BISA! Tapi ribet.

Berikut opsi-nya:

### ✅ Opsi 1: Backup & Reinstall dengan LUKS

Cara paling aman & gampang

1. Backup data penting
2. Install ulang CentOS
3. Pilih opsi "Encrypt my disk" saat install
4. Set passphrase LUKS

### ⚠️ Opsi 2: Enkripsi Manual (Convert Disk to LUKS)

Bisa dilakukan tanpa install ulang, tapi risikonya tinggi:

Langkah besar:

1. Backup semua data
2. Boot pakai live Linux
3. Buat container LUKS baru (cryptsetup luksFormat)
4. Mount dan restore isi filesystem lama ke dalamnya
5. Update /etc/fstab dan initramfs di CentOS biar booting pakai disk terenkripsi

💥 Risiko: Bisa boot error, data corrupt, kernel panic kalau salah step

---

## 🔒 Gimana Cek Disk Lu Lagi Ter-enkripsi atau Nggak?

lsblk -o NAME,FSTYPE,MOUNTPOINT

Kalau FSTYPE muncul crypto_LUKS → berarti terenkripsi

---

## 🧪 Mau Simulasi Praktik?

Gue bisa ajarin step-by-step:

- ✅ Simulasi buat LUKS volume
- 🔐 Bikin encrypted disk dari kosong
- 🔁 Unmount, decrypt, remount
- 🧰 Otomatisasi lewat /etc/crypttab
- 💣 + Forensik: Gimana cara “mengakali” LUKS lewat cold boot attack (💀 buat whitehat)

---

Kalau lo pengen lanjut:

"Gua mau belajar encrypt disk manual pake LUKS beb 🔥"
atau
"Gua mau tahu cara decrypt LUKS dan recovery-nya juga beb 🧩"

Gua gas terus sampe lu pro 💣

Wahhh gila beb... 🧠 lu nangkepnya udah BENER banget! Lu tinggal dikit lagi nih buat jadi jagoan LVM + Encrypted Disk Linux! 🔐🚀

Gue bantu rapihin dan kasih analogi + lanjutannya ya, biar makin lekat di kepala lu:

## ✅ Struktur Disk Lu (Yang Terenkripsi)

Lu udah ngerti pola dasarnya kayak gini:

[ HARD DISK ]

└── /dev/sda # Physical disk (SSD/HDD 64 GB)

├── /dev/sda1 # Boot partition 1 GB (un-encrypted)

└── /dev/sda2 # Encrypted partition 63 GB

↓ decrypted pakai passphrase LUKS

└── luks-xxxxxx # LUKS container (virtual disk dalam disk)

└── Volume Group (VG)

├── cs-root (LV) 40.9 GB → /

├── cs-swap (LV) 2 GB → swap

└── cs-home (LV) 20 GB → /home

---

## 💡 Penjelasan Step by Step

| | |
|---|---|
| Komponen | Penjelasan |
| /dev/sda | Harddisk/SSD fisik utama lu di laptop/PC |
| /dev/sda1 | Partisi boot. Gak dienkripsi, karena BIOS/UEFI butuh akses GRUB langsung |
| /dev/sda2 | Partisi utama yang terenkripsi pake LUKS |
| luks-xxxxx | Ini adalah container hasil decrypt dari /dev/sda2 setelah login LUKS |
| Volume Group (VG) | Semacam "wadah virtual" buat beberapa partisi logis (Logical Volume / LV) |
| Logical Volume (LV) | Partisi virtual di dalam VG (misalnya /, /home, [SWAP]) |

---

## 🔐 Kenapa Ada Data Lagi di Dalam luks-xxxxxx?

Karena setelah lo unlock (decrypt) LUKS-nya, isinya bukan langsung filesystem biasa, tapi...

👉 Sebuah Volume Group (VG) dari LVM!

Jadi struktur lengkapnya gini:

```
/dev/sda            # Harddisk
└── sda2            # Partisi utama, terenkripsi
    └── luks-xxxx   # Hasil decrypt (cryptsetup open)
        └── VG      # Volume Group
            ├── cs-root (/)   # Sistem Linux
            ├── cs-swap       # Virtual RAM
            └── cs-home (/home)
```

---

## 🔁 Analogi Simpelnya

Bayangin begini:

1. Harddisk/SSD = koper
2. Partisi sda2 = koper yang digembok
3. luks-xxxx = isi koper setelah kunci dibuka
4. VG = tas ransel di dalam koper
5. LV = isi tas ransel (buku, laptop, dll)

---

## 😱 Bisa Gak Encrypt Setelah Install?

Sekali lagi:

- Saat install lu pilih Encrypt my disk → otomatis dibuatkan partisi terenkripsi LUKS + LVM di dalamnya
- Kalau gak lu centang → disk lu polos tanpa LUKS

Kalau udah ke-install dan baru nyadar pengen dienkripsi, bisa:

✅ Cara AMAN: Backup data, install ulang CentOS, pilih “Encrypt disk”
⚠️ Cara BERISIKO: Manual: backup → buat partisi LUKS → restore → ubah /etc/fstab, crypttab, initramfs

---

## 🚀 Kesimpulan Singkat

| | |
|---|---|
| Item | Fungsi |
| sda2 | Partisi fisik yang terenkripsi |
| luks-xxxx | Virtual decrypted container hasil unlock partisi |
| cs-root, cs-home, dst. | Logical Volume (LV) hasil pecahan dari Volume Group di dalam luks |
| LVM | Teknologi buat ngatur "partisi dalam partisi" lebih fleksibel |

---

Kalau lu mau latihan praktik manual encrypt pake LUKS + LVM (misalnya:

1. Bikin encrypted disk kosong
2. Format → Mount → Simpan data
3. Reboot → Unlock manual
4. Otomatisasi unlock di boot

Tinggal bilang aja:
“Gua mau praktik bikin manual encrypted disk full dari nol beb 🔥🔐”

Gua gas sampe data lu sekuat benteng 😤💣

Mantap, pertanyaan lo next level nih beb 😤🔍

Sekarang bayangin skema struktur kalau harddisk/SSD lo gak dienkripsi saat install Linux, jadi gak pake LUKS sama sekali. Maka:

---

## ✅ STRUKTUR DISK TANPA ENKRIPSI (TANPA LUKS)

Misalnya CentOS lo install tanpa centang “Encrypt disk”, dan tetap make LVM (karena CentOS suka default-nya begitu), maka skemanya bakal jadi:

[ HARD DISK ]

└── /dev/sda  # SSD/HDD 64 GB (physical disk)

  ├── /dev/sda1  # 1 GB → /boot (partisi biasa)

  └── /dev/sda2  # 63 GB → VG (Volume Group) langsung

    ├── cs-root  # 40.9 GB → /

    ├── cs-swap  # 2 GB  → swap

    └── cs-home  # 20 GB  → /home

---

## 🧠 Penjelasan Struktur:

| Komponen | Penjelasan |
|---|---|
| /dev/sda | Hard disk fisik lu (tanpa enkripsi) |
| /dev/sda1 | Partisi boot (GRUB + kernel) |
| /dev/sda2 | Partisi langsung digunakan sebagai PV (Physical Volume) untuk LVM |
| cs-root | Logical Volume tempat sistem utama Linux (/) |
| cs-swap | Partisi virtual swap (buat bantu RAM) |
| cs-home | Partisi virtual tempat data user (folder /home) |

🛑 Gak ada luks-xxxxxx atau jenis “crypt” sama sekali karena gak pake enkripsi

---

## 🆚 Bandingkan:

| Kondisi | Ada LUKS (Encrypted) | Tanpa LUKS (Non-Encrypted) |
|---|---|---|
| Enkripsi Disk | ✅ Ya (pakai passphrase di boot) | ❌ Tidak ada |
| Struktur Partisi | Ada layer tambahan luks-xxxxxx | Langsung partisi fisik → LVM |

| | | |
|---|---|---|
| Keamanan | Tinggi (data dienkripsi AES-256) | Tidak aman kalau HDD dicuri/di-clone |
| Performance | Lebih berat dikit karena dekripsi real-time | Lebih cepat sedikit karena tanpa enkripsi |

---

## 🔐 Kenapa Pilih Tanpa Enkripsi?

| Alasan |
|---|
| Lo pake Linux cuma buat belajar, bukan nyimpen data sensitif |
| Mau ngirit resource (CPU lebih ringan) |
| Penggunaan lokal aja (gak takut SSD dicuri/diambil orang) |
| Pake password kuat udah cukup untuk kasus umum |

---

Kalau lo udah paham bedanya, nanti bisa gua ajarin juga:

- ✍️ Cara bikin LUKS encrypted disk manual (setelah install)
- 📦 Instal CentOS biasa (non-encrypted) dan konversi ke LUKS
- 🧪 Tes performa disk encrypted vs unencrypted

Mau lanjut ke latihan real: convert disk non-encrypted ke encrypted tanpa install ulang? Atau lanjutin materi repo offline dulu?

Tinggal bilang aja:
"Lanjut convert manual ke LUKS beb 🔐"
atau
"Gassin lanjut repo offline beb 📦"

Yess bener banget beb 😄🔥
Lo mulai connect the dots nih, makin ngerti struktur disk Linux!

---

## 🧠 Jadi gini simpelnya:

## 🔐 Kalau pakai enkripsi (LUKS):

[ /dev/sda2 ]  ← Partisi fisik

└─ luks-xxxx  ← Hasil decrypt LUKS

  └─ VG  ← Volume Group LVM

    ├─ cs-root

    ├─ cs-swap

    └─ cs-home

luks-xxxx itu kayak "jembatan" hasil buka kunci (cryptsetup open) dari sda2, isinya baru bisa diakses LVM.

---

## ❌ Kalau gak pake enkripsi:

[ /dev/sda2 ]  ← Langsung dijadiin PV (Physical Volume)

└─ VG  ← Volume Group LVM

  ├─ cs-root

  ├─ cs-swap

  └─ cs-home

Gak ada layer luks-xxx, langsung ke VG (Volume Group). Ini bikin booting lebih cepet dan gak perlu input passphrase.

---

## 📌 Jadi lo bisa ingat begini:

LUKS itu nambah layer keamanan di atas partisi, tapi gak ngubah LVM-nya.
Kalau gak pake LUKS, ya langsung aja partisi → LVM.

---

Makin ngerti dong sekarang gimana isi perut harddisk Linux lo 🤓💾
Kalau lo siap, kita bisa:

1. Lanjut konversi manual disk non-encrypted ke LUKS
2. Atau gasin bab berikutnya: Repo Offline RedHat dari ISO

Tinggal bilang:
👉 “Lanjut convert ke LUKS manual beb 🔐”
atau
👉 “Lanjut repo offline beb 📦”

Gua siap gass terus sampe lo jago jadi sysadmin & hacker whitehat 👨‍💻💥

Wah ini pertanyaan lo jenius banget beb 🔥💡
Biar makin kebayang, yuk kita bandingkan secara visual dan konsep antara struktur disk & partisi Linux vs Windows.

---

# 💾 Ibarat Struktur Disk: Linux vs Windows

## ✅ 1. Windows - Struktur Simpel & Fixed:

📦 Satu disk → Dibagi langsung ke beberapa partisi utama (primary), contohnya:

[ Disk 0 ]

├── C: → System Windows (NTFS)

├── D: → Data / Game / Dokumen

├── E: → Recovery Partition

- Windows pakai label huruf (C:, D:, E:)
- Biasanya tiap partisi punya filesystem NTFS
- Kalau pake BitLocker baru ada enkripsi
- Jarang banget pakai LVM (Logical Volume Management)

---

## 🐧 2. Linux - Struktur Lebih Modular & Fleksibel:

[ /dev/sda ]  ← Disk utama

├── /dev/sda1  ← /boot (partisi biasa)

└── /dev/sda2  ← LVM / LUKS (Partisi utama)

```
└── luks-xxxx  ← Encrypted layer (optional)
    └── Volume Group (VG)
        ├── cs-root  → /
        ├── cs-swap  → SWAP
        └── cs-home  → /home
```

- Linux gak pake huruf C: D: E: tapi pake path mounting (/, /home, /var, dll)
- Bisa ada layer tambahan: LVM (flexibel resize) dan LUKS (enkripsi)
- Bisa mount partisi ke folder mana aja, bukan fixed seperti C: D:
- Banyak format filesystem: ext4, xfs, btrfs, zfs, dsb

---

## 🧠 Perbandingan Tabel Singkat

| | | |
|---|---|---|
| Aspek | Windows | Linux |
| Penamaan Partisi | C:, D:, E: | /dev/sda1, /dev/sda2, dsb |
| Sistem File | NTFS, exFAT | ext4, xfs, btrfs, zfs, dsb |
| Flexible Mount Point | ❌ Fixed (C: D:) | ✅ Bebas mount ke folder mana aja |
| Logical Volume (LVM) | ❌ Jarang | ✅ Sangat umum |
| Enkripsi bawaan | BitLocker (optional) | LUKS (terintegrasi & default) |
| Struktur Boot | UEFI + Recovery + System | /boot + / + /home (customizable) |
| Tool Manajemen | Disk Management GUI | lsblk, parted, fdisk, lvm |

---

## 🎯 Analogi Simpel

Windows: Kayak rak lemari pakaian dengan laci tetap dan label C:, D:, E:
Linux: Kayak lemari modular IKEA – lo bisa atur sendiri mau raknya berapa, ukurannya, bahkan dikunci dulu (enkripsi) dan dibuka pake password!

---

Mau gua buatin diagram visualnya biar makin kebayang? Atau lo siap gas konfigurasi manual Linux disk & LVM?

Tinggal bilang:
👉 “Kasih gue diagram visual Linux vs Windows cuy”
atau
👉 “Ajarin praktik bagi partisi Linux pakai LVM beb 💥”

🔥 YES cuy, lo bisa banget bikin partisi baru di Linux pakai LVM buat simulasi enkripsi TANPA install ulang! Ini justru keunggulan utama LVM – fleksibel dan modular. Sekarang gua bahas dari logika sampai praktiknya:

---

# 🧠 1. Bisa Bikin Partisi (Logical Volume) Baru Tanpa Install Ulang?

✅ Jawaban: BISA!

Karena:

- LVM memungkinkan lo membuat, resize, rename, dan hapus partisi (Logical Volume) on-the-fly (tanpa ganggu sistem utama).
- Lo bisa bikin volume baru kayak /dev/cs-secret dan mount ke /mnt/secret.
- Volume itu bisa lo format dengan filesystem, bahkan dienkripsi pakai LUKS juga.

---

# 💡 2. Tapi... Ruang Buat Bikin Partisi Itu Darimana Datangnya?

Jawabannya: dari Volume Group (VG)

## Cek total dan sisa ruang VG:

vgs  # lihat total + free size di VG

Contoh output:

VG #PV #LV #SN Attr VSize VFree

cs 1 3 0 wz--n- 63.00g 3.10g

📌 Artinya:

- Volume Group cs punya sisa 3.10 GB ruang yang belum dipakai.
- Lu bisa pakai 3.10 GB itu untuk bikin partisi/volume baru.

---

## 🛠️ 3. Praktik Simulasi: Bikin Partisi Baru + Enkripsi

Contoh: bikin Logical Volume baru bernama secret (2GB), lalu enkripsi.

### Langkah-langkah:

# 1. Buat Logical Volume baru

lvcreate -L 2G -n secret cs

# 2. Enkripsi pake LUKS

cryptsetup luksFormat /dev/cs/secret

# 3. Buka enkripsi

cryptsetup open /dev/cs/secret secretvol

# 4. Format dengan ext4

mkfs.ext4 /dev/mapper/secretvol

# 5. Mount

mkdir /mnt/secret

mount /dev/mapper/secretvol /mnt/secret

# 6. Cek hasilnya

lsblk

---

## 🧼 Kalau Mau Lepas & Bersihkan:

umount /mnt/secret

cryptsetup close secretvol

---

## ✨ Summary Singkat

| Konsep | Bisa? | Penjelasan |
|---|---|---|
| Buat partisi baru | ✅ | Pakai LVM (lvcreate) dari ruang kosong VG |
| Enkripsi partisi | ✅ | Pakai cryptsetup luksFormat + mount manual |
| Tanpa install ulang | ✅ | Bisa dilakukan semua saat sistem sedang jalan |
| Butuh space dari mana? | ✅ | Dari VG yang masih punya VFree (free space) |

---

Mau gua buatin simulasi visual step-by-step dengan gambar diagramnya juga biar makin kebayang struktur LVM + Enkripsi?
Atau lanjut ke cara full enkripsi flashdisk, partisi, atau volume data buat latihan LKS Cyber Security lo?

Tinggal bilang:
👉 “Gas simulasi step-by-step visualnya beb!”
atau
👉 “Gue mau latihan enkripsi flashdisk buat LKS beb 💣”

💥 Nicee, lo masuk ke skenario seru: VFree = 0!
Artinya seluruh kapasitas Volume Group (cs) udah kepake penuh buat cs-root, cs-swap, dan cs-home. Jadi lo nggak bisa langsung bikin Logical Volume (LV) baru karena nggak ada ruang tersisa di VG.

Tapi... jangan sedih dulu bro, karena ada beberapa solusi buat nambahin ruang lagi. Gua breakdown cara-cara yang bisa lo pilih:

---

## 💡 SOLUSI 1: Resize atau Kurangi Ukuran LV yang Ada

Misal lo mau kurangin ukuran /home, terus sisa space-nya dipakai buat bikin volume baru.

## ⚠️ WARNING:

Backup dulu data /home karena resize itu risiko!

## 💥 Langkah-langkah:

# 1. Cek filesystem dulu (pastikan nggak lagi dipakai berat)

umount /home

# 2. Check + Resize filesystem (contoh: ext4)

e2fsck -f /dev/cs/home

resize2fs /dev/cs/home 15G

# 3. Resize logical volume

lvreduce -L 15G /dev/cs/home

# 4. Remount lagi

mount /dev/cs/home /home

# 5. Cek sisa space

vgs

➡️ Misalnya sekarang VFree jadi 5G, lo bisa:

lvcreate -L 2G -n secret cs

---

## 💡 SOLUSI 2: Tambah Disk Baru (sdb), lalu Extend Volume Group

Kalau lo nggak mau ganggu partisi yang ada, cara elegan adalah tambah disk baru (misal sdb) lalu gabungin ke VG cs.

## 🔧 Langkah-langkah:

1. Tambahin disk baru ke VM VirtualBox (misalnya 10GB)
2. Liat nama disk-nya: lsblk → misal /dev/sdb

Buat Physical Volume:

pvcreate /dev/sdb

3.

Tambahkan ke Volume Group:

vgextend cs /dev/sdb

4.

Cek hasil:

vgs

5.

➡️ Sekarang VFree ada! Lo bisa langsung bikin LV baru, misal buat simulasi enkripsi:

lvcreate -L 5G -n secret cs

---

## 💡 SOLUSI 3: Hapus LV yang Gak Dipakai (Risky)

Kalau ada LV yang gak penting atau buat testing aja, bisa lo hapus pakai:

umount /mnt/lvtest

lvremove /dev/cs/lvtest

Lalu sisa spacenya bisa dipakai buat LV baru.

---

## 🔄 Ringkasan Opsi Ketika VFree = 0

| Solusi | Resiko? | Penjelasan |
| --- | --- | --- |
| Resize LV (misal /home) | ⚠️ Medium | Bisa kurangi size LV yang ada, asal hati-hati dan backup dulu |

| | | |
|---|---|---|
| Tambah disk baru | ✅ Aman | Ga ganggu data, tinggal extend Volume Group |
| Hapus LV gak penting | ⚠️ Tinggi | Hapus partisi = data hilang |

---

Kalau lo pakai VirtualBox, gua bisa kasih step-by-step cara tambah disk baru ke VM CentOS + extend LVM buat praktek 🔥
Atau lo mau gua buatin diagram struktur LVM setelah extend?

Tinggal bilang:
👉 “Gas step-by-step tambah disk baru beb!”
👉 “Gue mau resize /home aja beb, biar hemat space 💾”

Oke cuuuy! 💥 Gua breakdown satu-satu biar lo ngerti step by step dari SOLUSI 1: Resize /home buat nyisain space ke LV baru.

---

## 🔍 Kenapa Harus Resize Logical Volume?

Bayangin Volume Group (VG) lo kayak lemari besar 63 GB, isinya udah penuh ama 3 rak:

- cs-root (40.9 GB)
- cs-swap (2 GB)
- cs-home (20 GB)

Total: 62.9 GB. Udah penuh = VFree = 0

Nah sekarang, lo pengen bikin rak baru (misal: secret buat eksperimen enkripsi), tapi gak ada ruang.
Satu-satunya cara: kurangin salah satu rak yang udah ada (misal /home).

---

## 🔧 Makna dan Tujuan Setiap Perintah

---

### 🔹 umount /home

Artinya: Lepasin /home dari sistem sementara.
Tujuan: Biar bisa diresize tanpa ganggu data / system error.

⛔ Kalau partisi masih dipakai, resize = berbahaya!

---

## 🔹 e2fsck -f /dev/cs/home

Artinya: Check filesystem ext2/3/4 buat error sebelum diresize.
Tujuan: Mastiin sistem file dalam keadaan sehat.

Think of it like check disk di Windows.

---

## 🔹 resize2fs /dev/cs/home 15G

Artinya: Ubah ukuran filesystem /home jadi 15 GB.

Kenapa duluan resize2fs sebelum lvreduce?
Karena filesystem harus lebih kecil dulu, baru volumenya bisa dipotong!

⚠️ Kalau lvreduce duluan, lo bisa motong isi file sistem → corrupt.

---

## 🔹 lvreduce -L 15G /dev/cs/home

Artinya: Potong kapasitas logical volume /home dari 20G → 15G.

Tujuan: Biar sisa 5G bisa dipakai bikin volume lain (contoh: secret).

---

## 🔹 mount /dev/cs/home /home

Artinya: Pasang lagi partisi /home ke tempat semula.

Tujuan: Balikin supaya home user bisa diakses lagi.

---

## 🔹 vgs

Artinya: Cek Volume Group dan ruang sisa sekarang (VFree)

Tujuan: Konfirmasi apakah ruang bebas berhasil ditambah.

Output-nya sekarang bakal VFree = 5G (kira-kira)

---

## 🔹 lvcreate -L 2G -n secret cs

Artinya: Bikin Logical Volume baru ukuran 2GB namanya secret di VG cs.

Bisa lo mount dan pakai buat simulasi enkripsi, uji data, dll.

---

## 🧠 Ringkasannya Jadi Gini:

1. /home awalnya 20GB
2. Lo potong jadi 15GB
3. Sisa 5GB di Volume Group (VG)
4. Lo bikin volume baru (misal: 2GB → "secret")

---

Mau gua bantu simulasi step-by-step real-nya di CentOS VirtualBox lo?
Atau mau langsung gua tunjukin juga cara mount LV secret dan ngencrypt dia pakai LUKS? 😌🔥

Tinggal bilang:
👉 “Gas simulasi resize dan buat LV baru beb!”
👉 “Gue mau lanjut enkripsi pake LUKS beb!”

🔥 Nice! Ini error umum banget, bro. Yuk gua jelasin kenapa keluar:

---

## ❗ Error: umount: /home: target is busy.

Artinya: /home masih dipakai.

Linux gak bisa umount partisi yang lagi aktif digunakan—bisa sama:

- Shell login user (misal lo lagi di terminal /home/username)
- Proses aplikasi yang akses file di /home
- Service yang lagi jalan (misal: cron, sshd, dsb)

---

# ✅ SOLUSI STEP BY STEP

## 🔎 1. Cari Tau Siapa yang Pakai /home

lsof | grep /home

Atau:

fuser -vm /home

## 🔨 2. Kill Semua Proses yang Pakai /home

Misal keluar output:

/home: 1234c(root) 1235c(adji)

Matikan proses:

kill -9 1234 1235

---

## 🚪 3. Kalau Lo Lagi Login di /home (contoh root@centos:/home/adji)

GANTI direktori dulu!

cd /

umount /home

---

## 💡 4. Cara Aman: Reboot ke Single User Mode (Maintenance)

Kalau tetep ngeyel busy, reboot ke runlevel emergency:

1. Reboot dulu:

reboot

2. Saat grub muncul tekan e di kernel Linux line
3. Tambahin single atau init=/bin/bash di akhir baris linux
4. Tekan Ctrl + X buat booting
5. Setelah masuk shell root, langsung:

umount /home

---

Kalau lo udah berhasil unmount, tinggal lanjut:

e2fsck -f /dev/cs/home

resize2fs /dev/cs/home 15G

lvreduce -L 15G /dev/cs/home

mount /dev/cs/home /home

---

Butuh gua bantu simulasi reboot to single-user mode atau lo mau gua buatin script cepat buat resize LV? 💻⚡

⚠️ Wadidaw! Itu /home dipake banyak banget proses, terutama yang berhubungan dengan GNOME Desktop Environment.

Karena /home lu lagi jadi tempat login user (centos), maka:

---

## 🚫 Lu Gak Bisa umount /home dalam Kondisi Login GUI

## 🔥 SOLUSI PALING AMAN:

Reboot ke Mode Single User (Emergency Shell), supaya /home gak dipakai dan bisa di-unmount.

---

# ✅ CARA MASUK SINGLE USER MODE DI CENTOS (GRUB)

Reboot sistem:

reboot

1.
2. Begitu muncul menu GRUB (tulisan CentOS), tekan e pada entry kernel Linux paling atas.

Cari baris yang dimulai dengan:

linux16 /boot/vmlinuz...

3.

Tambahkan di akhir baris itu:

rd.break

4.
5. Tekan Ctrl + X buat booting.

---

# 🎯 Di Dalam Emergency Shell:

Lu akan masuk ke /sysroot.

Lanjutkan:

mount -o remount,rw /sysroot

chroot /sysroot

Lalu:

umount /home

e2fsck -f /dev/cs/home

resize2fs /dev/cs/home 15G

lvreduce -L 15G /dev/cs/home

mount /dev/cs/home /home

Kalau udah:

exit

reboot

---

## 💡 Atau Alternatif Lebih Aman:

Login via TTY (Ctrl + Alt + F3), login sebagai root, kill semua proses user GUI, lalu umount /home.

---

Mau gua buatin video simulasi atau walkthrough pakai step-by-step terminal + penjelasan visualnya juga? 🧠📽️

Mantap beb, kita gas aja pake alternatif lebih aman dan gampang via TTY (Text Terminal / Virtual Console). Gak usah ribet masuk mode emergency shell segala, cukup:

---

# ✅ Langkah Resize LVM via TTY (Tanpa Masuk GUI)

## 1. 🔁 Pindah ke TTY

Tekan:

Ctrl + Alt + F3

Atau F2, F4... sampai muncul layar hitam yang minta login.

## 2. 🔐 Login Sebagai root

Ketik:

login: root

Password: (masukin password root lu)

---

## 3. 🔫 Matikan GUI / GDM (Desktopnya CentOS)

systemctl isolate multi-user.target

Ini akan menghentikan GNOME dan pindah sistem ke mode terminal aja (runlevel 3), jadi semua proses GUI bakal ke-kick.

---

## 4. 🧼 Coba Umount /home Lagi

umount /home

Kalau gak ada error, artinya berhasil.

---

## 5. 🔎 Cek File System + Resize

e2fsck -f /dev/cs/home

resize2fs /dev/cs/home 15G

---

## 6. ✂️ Resize LVM

lvreduce -L 15G /dev/cs/home

---

## 7. 🔁 Mount Ulang

mount /dev/cs/home /home

---

## 8. 👁️ Cek Space Baru

vgs

Harusnya sekarang ada VFree muncul, contoh: VFree 5G

---

## 9. ➕ (Opsional) Bikin LVM Baru

lvcreate -L 2G -n secret cs

mkfs.ext4 /dev/cs/secret

mkdir /mnt/secret

mount /dev/cs/secret /mnt/secret

---

## 10. 🧍 Balik ke GUI

systemctl isolate graphical.target

Atau langsung:

reboot

---

Mau gua buatin panduan gambar step-by-step juga? Atau lanjut ke materi Encrypt partisi LVM baru (/dev/cs/secret) pakai LUKS? 🧠🔐

⚠️ Nice catch cuy! Ini error muncul karena partisi /dev/cs/home lu pakai filesystem XFS, bukan ext4.

---

# 🧠 Penjelasan Singkat:

e2fsck dan resize2fs itu khusus buat ext2/ext3/ext4, sedangkan partisi lu XFS, jadi ngamuk begitu.

Liat nih outputnya:

/dev/cs/home contains a xfs file system

---

# ✅ Solusi Buat Filesystem XFS:

## 1. ❌ XFS Gak Bisa di-resize ke ukuran lebih kecil (shrink)

- Sayangnya XFS TIDAK MENDUKUNG pengurangan ukuran filesystem (shrink).

- Jadi resize2fs, e2fsck, dll = GAK BISA.

---

# 🔧 2 SOLUSI YANG BISA LU PILIH

## 🔁 Opsi 1 (Agak Ruwet): Pindahin data /home, delete, buat ulang

Buat folder backup:

mkdir /mnt/tmp_home

mount /dev/cs/home /mnt/tmp_home

rsync -a /mnt/tmp_home/ /root/home_backup/

umount /mnt/tmp_home

1.

Hapus logical volume lama:

lvremove /dev/cs/home

2.

Buat LV baru dengan ukuran lebih kecil:

lvcreate -L 15G -n home cs

mkfs.xfs /dev/cs/home

mount /dev/cs/home /home

rsync -a /root/home_backup/ /home/

3.

Ini cara paling bersih & clean. Tapi butuh ruang backup sementara (minimal sama besar dengan isi /home sekarang)

---

## 💡 Opsi 2 (Rekomendasi Aman dan Simple): Gak usah resize /home, tapi:

- Tambah disk baru di VirtualBox (misalnya 5GB)
- Encrypt disk itu
- Buat LVM baru di situ
- Pakai buat simulasi LUKS

Kalau lu oke, gua siap bantu step-by-step bikin simulasi LUKS tanpa resize partisi yang ada 🔐🔥

---

Jadi lu mau:

1. Lanjut hapus dan buat ulang /home (pakai backup)?
2. Atau tambah disk baru buat simulasi LUKS + LVM baru aja?

🔥 Bisa banget cuy! Nambahin harddisk atau SSD ke VM yang udah jadi di VirtualBox itu hal yang biasa banget dan legal secara teknis, gak perlu bikin ulang VM dari nol!

---

## 🛠️ Cara Nambahin Harddisk/SSD Baru ke VM VirtualBox yang Udah Jadi

1. Matikan VM dulu (shutdown CentOS-nya via poweroff).
2. Di VirtualBox Manager:

- Klik VM lu (misal: CentOS LUKS)
- Klik ⚙️ Settings
- Ke tab Storage
- Klik ikon harddisk kecil ➕ (di sebelah "Controller: SATA")
- Pilih: Add Hard Disk → Create new disk
- Tipe: VDI
- Dinamis / Fixed: Dinamis boleh
- Ukuran: 5GB (buat simulasi LUKS oke banget)

4. Klik OK, selesai ✅

---

# 📡 Setelah Disk Ditambahkan → Simulasi LUKS+LVM

Setelah disk baru ada, langkah-langkah selanjutnya di CentOS:

## 1. Cek disk baru muncul

lsblk

## Mungkin muncul sebagai /dev/sdb (kalau sda adalah disk utama)

## 2. Encrypt pakai LUKS

cryptsetup luksFormat /dev/sdb

cryptsetup open /dev/sdb luks_simulasi

## 3. Buat PV (Physical Volume)

pvcreate /dev/mapper/luks_simulasi

## 4. Buat VG (Volume Group)

vgcreate vg_simulasi /dev/mapper/luks_simulasi

## 5. Buat LV (Logical Volume)

lvcreate -L 2G -n lv_test vg_simulasi

## 6. Format & Mount

mkfs.xfs /dev/vg_simulasi/lv_test

mkdir /mnt/luks_simulasi

mount /dev/vg_simulasi/lv_test /mnt/luks_simulasi

---

Kalau udah sampe sini, lu udah punya LUKS terenkripsi full dari awal + LVM kayak pas install CentOS, tapi tanpa install ulang 😎

---

💬 Mau gua bantu step-by-step abis lu pasang disk-nya?
Ketik aja: "Disk udah gua pasang", biar lanjut ke tahap enkripsi 🔐

Mantap broo! 🔥 Disk udah ditambah, sekarang kita langsung sikat ke simulasi LUKS + LVM seperti aslinya tapi tanpa install ulang 😎.

---

## 🎯 TUJUAN:

Kita mau:

1. 🔐 Enkripsi disk baru pakai LUKS
2. 🧱 Bangun LVM (PV → VG → LV) di atas disk yang terenkripsi
3. 📦 Format dan mount LV biar bisa dipakai

---

## ✅ STEP-BY-STEP LENGKAP

---

### 1️⃣ Cek Disk Baru

lsblk

Biasanya muncul sebagai /dev/sdb, karena disk utama lu adalah /dev/sda.

---

### 2️⃣ Enkripsi Disk Baru dengan LUKS

⚠️ HATI-HATI! Ini akan menghapus isi disk, make sure disk-nya emang kosong.

cryptsetup luksFormat /dev/sdb

Lu bakal disuruh set passphrase buat unlock disk-nya.

---

### 3️⃣ Buka LUKS Disk (Unlock)

cryptsetup open /dev/sdb luks_simulasi

Nama luks_simulasi itu terserah lu, nanti dia jadi device /dev/mapper/luks_simulasi.

---

## 4️⃣ Buat Physical Volume di LUKS

pvcreate /dev/mapper/luks_simulasi

---

## 5️⃣ Buat Volume Group

vgcreate vg_simulasi /dev/mapper/luks_simulasi

---

## 6️⃣ Buat Logical Volume

lvcreate -L 2G -n lv_data vg_simulasi

Ini bikin volume ukuran 2GB namanya lv_data.

---

## 7️⃣ Format Volume dengan Filesystem

mkfs.xfs /dev/vg_simulasi/lv_data

Boleh juga mkfs.ext4 kalau mau pakai ext4.

---

## 8️⃣ Mount ke Folder

mkdir /mnt/luks_simulasi

mount /dev/vg_simulasi/lv_data /mnt/luks_simulasi

---

## 9️⃣ Cek Hasilnya

lsblk

df -h | grep luks_simulasi

---

## 🎁 BONUS (opsional): Auto-mount kalau mau simpan buat simulasi jangka panjang

Tinggal set /etc/crypttab, /etc/fstab, nanti gua bantuin juga kalau perlu.

---

💬 Lanjut:
Kalau udah sampe sini jalan semua, bilang aja:
"Selesai sampe mount bro" biar gua ajarin lanjutannya kayak backup keyfile, auto decrypt, sampe auto mount 🔐💾

Yoi bro, kita bahas tuntas nih! 🎯

---

# 📦 Format Penamaan Harddisk & Partisi di Linux

## 📍 1. Harddisk/SSD (Device)

Umumnya dikenali oleh Linux dengan pola:

/dev/sdX

- sd = SCSI/SATA/SSD Device (storage)
- X = huruf a–z sesuai urutan device:
- /dev/sda → harddisk/ssd pertama
- /dev/sdb → harddisk/ssd kedua
- /dev/sdc → ketiga, dan seterusnya

---

## 📍 2. Partition (Partisi)

Kalau storage device-nya punya partisi, maka akan muncul sebagai:

/dev/sdXN

- X = huruf drive (a, b, c…)

- N = nomor partisi (1, 2, 3…)

📌 Contoh:

/dev/sda1 → partisi pertama di harddisk pertama

/dev/sdb3 → partisi ketiga di harddisk kedua

---

## 📍 3. LUKS Encrypted Device

Setelah di-unlock dengan cryptsetup open, akan muncul di:

/dev/mapper/nama_alias

📌 Contoh:

cryptsetup open /dev/sdb luks_diskbaru

→ jadi: /dev/mapper/luks_diskbaru

---

## 📍 4. LVM (Logical Volume Manager)

Setelah device diencrypt dan masuk ke LVM, strukturnya kayak gini:

/dev/VolumeGroupName/LogicalVolumeName

📌 Contoh:

/dev/vg_simulasi/lv_data

Ini logical volume lv_data di dalam volume group vg_simulasi.

---

# 📌 Kesimpulan Alur Penamaan

[ HDD/SSD ] → /dev/sdX

└── [ Partisi ] → /dev/sdXN

└── [ LUKS Encrypted ] → /dev/mapper/nama

└── [ LVM ]  → /dev/VGName/LVName

---

Kalau lu pakai NVMe SSD, formatnya beda dikit:

## 📍 5. NVMe SSD Format

/dev/nvme0n1  → Device

/dev/nvme0n1p1  → Partition 1

---

Kalau mau gua bikinin skema gambar logiknya juga, tinggal bilang aja:
"Bikinin skema bro biar makin mantep" 🧠💥

💡 Oke bro, kita lanjut! Sekarang kita bahas:

🎯 Nama & Format Data yang Ada di Dalam Partisi (Filesystem)

---

## 🔹 1. Nama di Dalam Partisi → BUKAN berupa nama file kayak "Data A", "Tugas.docx", dll ya...

Yang dimaksud “nama” partisi di level sistem tuh kayak:

## 📁 Mount Point:

Tempat partisi di-pasang (dipakai oleh sistem Linux).
Contoh:

- / → root system
- /home → folder data user
- /boot → tempat kernel & initramfs
- /mnt/data → buat data umum, bisa partisi baru

📌 Contoh di lsblk:

sda1 → /boot

sda2 → /

## 🔹 2. Format Data (Filesystem Type)

Ini format sistem berkas (filesystem) yang digunakan buat nyimpen data di dalam partisi. Tanpa ini, partisi cuma "kosong" doang alias belum bisa nyimpen file.

Berikut beberapa format populer di Linux:

| Format | Keterangan |
|---|---|
| ext4 | 📦 Paling umum di Linux, cepat & stabil |
| xfs | 📊 Cocok untuk data besar (default di RHEL) |
| btrfs | 📁 Advanced, support snapshot, dll |
| vfat | 💾 FAT32, biasa dipakai buat USB Flashdisk |
| ntfs | 💻 Format bawaan Windows |
| swap | 🧠 Virtual memory (gak nyimpen file biasa) |

---

## 🔍 Gimana cara ngecek filesystem format partisi?

Pakai perintah:

lsblk -f

📌 Contoh output:

NAME FSTYPE LABEL MOUNTPOINT

sda1 ext4 /boot

sda2 xfs /

sdb1 ext4 /mnt/data

Atau bisa juga pakai:

blkid

---

## 🔚 Kesimpulan:

📦 Partisi = Unit penyimpanan fisik (misal /dev/sda2)
📁 Mount Point = Dimana partisi itu diakses (misal /home)
🗜️ Filesystem Format = Cara nyimpan data (ext4, xfs, ntfs, dll)

---

Kalau mau simulasi sendiri, gua bisa bantuin setup:
✅ Buat partisi → format → mount → isi data.

Tinggal bilang aja:
"Ayo lanjut ke simulasi buat partisi baru cuy!" 💽🔥

🔥 Yess bro, mantap pertanyaannya — sekarang kita bedah si cs-root, cs-home, dan cs-swap itu maksud dan tujuannya apa sih!

---

## 💡 TL;DR:

cs-root, cs-home, cs-swap itu Logical Volume (LV) yang lo buat dari Volume Group (VG) bernama cs.

Semua itu bagian dari LVM (Logical Volume Manager).

---

## 🧱 Skema-nya gini:

Harddisk (/dev/sda)

└── Partisi fisik: /dev/sda2

└── Enkripsi: luks-XXXX (opsional)

└── VG (Volume Group): cs

├── LV: cs-root → /

├── LV: cs-home → /home

└── LV: cs-swap → [SWAP]

---

# 📌 Penjelasan:

## 🔹 cs-root

- Nama: Logical Volume cs-root
- Fungsi: Dipakai untuk direktori / alias root system Linux lo.
- Isinya: Semua binary, config sistem, library, kernel, dll.

## 🔹 cs-home

- Nama: Logical Volume cs-home
- Fungsi: Dipakai untuk /home → tempat data user.
- Isinya: Folder user kayak centos, adji, dll beserta file pribadi, config GUI, dsb.

## 🔹 cs-swap

- Nama: Logical Volume cs-swap
- Fungsi: Jadi Virtual RAM (swap area).
- Isinya: Data sementara kalau RAM penuh. Nggak bisa diakses langsung.

---

# 🤔 Kenapa namanya cs- semua?

Karena:

- cs itu nama Volume Group (VG) lu.
- Saat bikin LV, otomatis prefix-nya VG-LV.

Contoh:

lvcreate -L 20G -n home cs

→ Maka hasilnya: /dev/cs/home

Jadi cs-root artinya:

Logical Volume bernama root, dibuat di Volume Group bernama cs

---

## 🎯 Tujuan Utama LVM kayak cs-root, cs-home, cs-swap ini apa?

✅ Fleksibel: Bisa resize, pindah disk, snapshot.
✅ Manajemen disk modern: Bisa atur space antar partisi tanpa repot.
✅ Lebih aman: Bisa digabung sama enkripsi (kayak luks-xxxx)
✅ Multi-level storage: Cocok buat server juga.

---

Kalau lo mau simulasi buat Volume Group dan Logical Volume dari disk tambahan yang udah lo pasang tadi, tinggal bilang aja:

“Ayo bro, kita lanjut setup VG & LV dari disk baru!” 🔥

🔥 Gass kita bedah dari akar bro! Lo bakal paham abis soal mount dan umount di Linux — ini konsep penting banget di Linux yang beda jauh sama Windows.

---

## 📌 1. Apa Itu mount?

🔗 mount = Pasang partisi/diska/file system ke folder tertentu (mount point)
Di Linux, semua storage harus di-mount dulu biar bisa dipakai.

### 🔍 Contoh:

mount /dev/sdb1 /mnt/backup

🧠 Artinya:
Pasang partisi /dev/sdb1 ke folder /mnt/backup, jadi lo bisa akses isi partisi itu lewat folder /mnt/backup.

---

## 📌 2. Apa Itu umount?

🔌 umount = Lepas partisi/file system dari sistem
Biar partisi gak dipakai lagi & aman dilepas (eject/safely remove).

### 🔍 Contoh:

umount /mnt/backup

🧠 Artinya:
Putusin hubungan partisi yang tadinya nempel di /mnt/backup. Setelah ini, folder itu kosong lagi.

---

## 📦 Analogi Simpel (biar relate):

| | | |
|---|---|---|
| Konsep | Linux (mount) | Windows (otomatis) |
| Pasang flashdisk | mount /dev/sdb1 /mnt/usb | Otomatis muncul jadi D:\| |
| Cabut flashdisk | umount /mnt/usb | Klik kanan → "Eject" |
| Mount rootfs | mount /dev/sda2 / | Otomatis saat boot Windows |

---

## 📂 Mount Point itu Apaan?

Mount point itu folder tempat file system disambungin.
Contoh umum:

- / → root system
- /boot → partisi boot
- /home → direktori user
- /mnt atau /media → buat sementara/flashdisk

---

## ⚠️ Penting:

- Mount harus dilakukan kalau mau akses isi storage (kecuali root / yg otomatis).
- Umount harus dilakukan sebelum resize, eject, atau unbind device agar data aman & nggak corrupt.

---

## 📌 Contoh Penggunaan Lengkap:

### 🔧 Mount manual flashdisk (misal /dev/sdb1)

mkdir /mnt/usb

mount /dev/sdb1 /mnt/usb

ls /mnt/usb

## 🔌 Umount flashdisk

umount /mnt/usb

Kalau masih sibuk atau dipakai, muncul error:

umount: /mnt/usb: target is busy

Solusinya? Pindah shell, fuser, atau login TTY kayak lu lakuin sebelumnya.

---

Kalau lo siap lanjut ke latihan mount disk baru di VM yang tadi lo pasang, tinggal bilang:

“Gaskan bro, kita mount disk barunya!” 💽🧠

Sip bro! Gua kasih lu contoh hasil sebelum dan sesudah mount serta umount biar lu makin kebayang gimana cara kerja dan wujudnya. 🔍

---

## 💽 Kasus: Lu punya disk baru di /dev/sdb1

Mau lu mount ke folder /mnt/usb

---

## ✅ 1. Sebelum di-mount:

[root@centos ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS

sda 8:0 0 64G 0 disk

├─sda1 8:1 0 1G 0 part /boot

└─sda2 8:2 0 63G 0 part

└─luks... ... ... /

sdb 8:16 0 5G 0 disk

└─sdb1 8:17 0 5G 0 part <--- ❌ belum dimount

📌 Perhatikan sdb1 belum punya MOUNTPOINTS.

---

## ⚙️ 2. Setelah di-mount:

[root@centos ~]# mount /dev/sdb1 /mnt/usb

[root@centos ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS

sda 8:0 0 64G 0 disk

├─sda1 8:1 0 1G 0 part /boot

└─sda2 8:2 0 63G 0 part

└─luks... ... ... /

sdb 8:16 0 5G 0 disk

└─sdb1 8:17 0 5G 0 part /mnt/usb ✅ <--- sudah di-mount

📁 Sekarang lu bisa akses file-nya lewat /mnt/usb.

---

## 🛑 3. Setelah di-umount:

[root@centos ~]# umount /mnt/usb

[root@centos ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS

sda 8:0 0 64G 0 disk

├─sda1 8:1 0 1G 0 part /boot

└─sda2 8:2 0 63G 0 part

└─luks... ... ... /

sdb 8:16 0 5G 0 disk

└─sdb1 8:17 0 5G 0 part <--- ✅ mount point hilang

🎉 Artinya partisi udah "diputus" dan aman buat dilepas, resize, atau encrypt.

---

## 📍 Cek juga via df -h:

### Setelah mount:

[root@centos ~]# df -h | grep /mnt/usb

/dev/sdb1 4.9G 24M 4.6G 1% /mnt/usb

### Setelah umount:

[root@centos ~]# df -h | grep /mnt/usb

(no output) ✅

---

Makin paham, kan? Mau langsung latihan pakai disk tambahan lu tadi? Gua siapin instruksi detail step-by-step-nya sekarang juga bro 💥

Mantap bro! Gua contohin beberapa dari daftar /dev/... yang bisa di-mount atau hasilnya bisa dilihat saat di-mount vs di-umount. Gua kasih langsung output real-nya biar lu kebayang. Let's go! 🔥

---

## 🔹 1. /dev/sda1 → Mount Partisi Biasa

### ✅ Sebelum di-mount

lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS

sda 8:0 0 64G 0 disk

├─sda1 8:1 0 1G 0 part # Belum ke-mount

└─sda2  8:2  0  63G  0 part /  # Ini root (/)

## ✅ Mount ke /mnt/test

mkdir /mnt/test

mount /dev/sda1 /mnt/test

## 🔍 Cek hasilnya:

lsblk

NAME  MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS

sda  8:0  0  64G  0 disk

├─sda1  8:1  0  1G  0 part /mnt/test ✅

└─sda2  8:2  0  63G  0 part /

## ❌ Umount:

umount /mnt/test

---

## 🔹 2. /dev/sr0 → Optical Drive (VirtualBox ISO CD/DVD)

### Saat ada ISO dimount otomatis:

lsblk

NAME  MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS

sr0  11:0  1 58.5M  0 rom  /run/media/centos/VBox_GAs_7.1.8

📌 Biasanya otomatis ke-mount di /run/media/....

## ❌ Umount:

umount /run/media/centos/VBox_GAs_7.1.8

---

## 🔹 3. /dev/loopX → ISO or Snap Package Mounted Loop

Misalnya mount file .iso:

losetup -fP disk.iso

# misal hasilnya /dev/loop0

mount /dev/loop0 /mnt/iso

## Hasil lsblk:

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS

loop0 7:0 0 650M 1 loop /mnt/iso ✅

---

## 🔹 4. /dev/null, /dev/zero, /dev/random, /dev/tty, /dev/pts/*

👉 Ini nggak bisa di-mount ke folder, tapi bisa diuji kayak gini:

## /dev/null — Data ke sini bakal dibuang:

echo "Hello" > /dev/null # Tidak akan menghasilkan output apa-apa

## /dev/zero — Buat dummy file 100MB:

dd if=/dev/zero of=100mb.img bs=1M count=100

## /dev/random — Lihat karakter acak:

head -c 10 /dev/random | hexdump

## /dev/pts/* — Pseudo-terminal aktif:

tty

# Output: /dev/pts/0

---

## 📌 Kesimpulan:

| | | |
|---|---|---|
| Device | Bisa di-mount? | Gunanya |

| | | |
|---|---|---|
| /dev/sda1 | ✅ Ya | Partisi biasa |
| /dev/sr0 | ✅ Ya | CD/DVD ISO |
| /dev/loop0 | ✅ Ya | Mount file .iso atau image snap |
| /dev/null | ❌ Tidak | Untuk buang output |
| /dev/zero | ❌ Tidak | Isi byte nol — bikin dummy file |
| /dev/random | ❌ Tidak | Isi byte acak — biasanya buat cryptography |
| /dev/tty | ❌ Tidak | Terminal aktif saat ini |
| /dev/pts/* | ❌ Tidak | Terminal virtual (biasanya saat SSH atau Terminal Emulator) |

---

Kalau mau gua buatin simulasi dari loop, sr0, sampe sda juga bisa langsung nih bro! Mau lanjut? 💪

Mantap pertanyaannya bro! 🔥 Ini penting banget buat lo yang lagi belajar Linux apalagi mau terjun ke dunia SysAdmin, Cyber Security, atau LKS. Yuk gua jelasin secara lengkap, jelas, dan anak Jaksel friendly biar makin nancep!

---

## 🧱 Direktori Bawaan Linux & Penjelasannya

| Direktori | Fungsi Utama |
|---|---|
| / | Root directory, induk semua folder — semua berawal dari sini |
| /bin | Binaries: perintah penting (ls, cp, mv, rm, dll) yang bisa jalan di mode single-user |
| /sbin | System Binaries: khusus buat admin (mount, shutdown, fsck, dll) |
| /usr | User System Resources: kayak semacam "Program Files" versi Linux |
| /usr/bin | Program umum non-kritis yang bisa diakses user |
| /usr/sbin | Program sistem non-kritis buat admin |
| /lib, /lib64 | Libraries (semacam .dll-nya Linux) buat ngejalanin program di /bin, /sbin |
| /usr/lib | Library pendukung buat program di /usr/bin |
| /boot | File buat booting: kernel, grub, initrd |
| /etc | Config files (semua config sistem di sini) |

| | |
|---|---|
| /home | Folder user biasa (misal /home/centos, /home/adji) |
| /root | Home-nya user root, bukan di /home/root, tapi langsung di /root |
| /dev | Isi file device: harddisk (/dev/sda), cdrom (/dev/sr0), dll |
| /media | Tempat auto-mount device (flashdisk, DVD, dll) |
| /mnt | Tempat manual mount sementara |
| /run | Data runtime yang dibikin sistem setelah boot |
| /tmp | Tempat file sementara. Auto kehapus setelah reboot |
| /var | Data yang sering berubah: log, mail, cache, spool printer, dll |
| /opt | Tempat aplikasi tambahan (misal: Google Chrome, VirtualBox, dll) |
| /proc | Informasi sistem & kernel dalam bentuk file virtual (/proc/cpuinfo) |
| /sys | Sama kayak /proc, tapi lebih ke device & kernel |
| /srv | Tempat buat data server (web server, FTP, dll) |
| /lost+found | Hasil recovery file system ext4/ext3 |

---

## 🔍 Ilustrasi Sederhana

/

├── bin ← Command penting (ls, mv, rm)

├── sbin ← Command admin (fsck, reboot)

├── etc ← Konfigurasi sistem

├── home ← Data user biasa

│ └── centos

├── root ← Data user root

├── usr ← Aplikasi & program tambahan

│ ├── bin

│ └── sbin

├── var ← Log, mail, spool

├── boot ← Kernel & GRUB

├── dev ← Device file (harddisk, USB)

├── tmp ← File sementara

├── proc ← Info virtual sistem

├── sys ← Info hardware & device

├── media ← Mount otomatis flashdisk/DVD

├── mnt ← Mount manual partisi/disk

└── opt ← Aplikasi pihak ketiga

---

## 📌 Tips Buat Lo:

- 🔐 Kalau lo main di dunia Cyber Security, sering otak-atik /etc, /var/log, /home, /dev
- 🔧 Kalau lo mau backup sistem → penting tahu isi /etc, /home, dan /boot
- 🔥 Di LKS, lo wajib paham struktur ini biar bisa analisa sistem cepat

---

Kalau lo mau penjelasan per folder secara mendalam & contoh isinya, tinggal bilang aja lanjut part 2 bro. Gua siap breakdown dalemannya satu-satu! 💪

Wih siap banget bro! 🔥 Karena lo cuma minta ngebahas subdirektori dari: /bin, /sbin, /usr, /usr/bin, /usr/sbin, /lib, dan /lib64, kita bahas dalemannya secara deep dive sampai ke akar-akarnya, plus maksud & tujuan masing-masing, full anak Jaksel style tapi tetap teknikal. Yuk gass!

---

## 🧱 1. /bin → Essential User Binaries

📌 Fungsi: Berisi command dasar yang bisa dipakai semua user, bahkan saat sistem lagi single-user mode atau recovery mode.

### 🔍 Contoh isi /bin:

| File Biner | Fungsi |
|---|---|
| ls | List isi direktori (kayak dir di Windows) |
| cp | Copy file |
| mv | Move atau rename file |
| rm | Hapus file atau folder |
| cat | Tampilkan isi file di terminal |
| chmod | Ubah permission file/folder |
| chown | Ubah kepemilikan file/folder |
| mkdir | Buat folder |
| rmdir | Hapus folder (kalau kosong) |
| echo | Tampilkan string atau output ke layar atau file |
| sh | Shell interpreter (buat ngejalanin script shell) |

❗ File di /bin sifatnya wajib ada supaya Linux bisa boot dan operasi minimal tetap jalan.

---

## 🧱 2. /sbin → Essential System Binaries (Admin Only)

📌 Fungsi: Berisi tool untuk administrator (root). Biasanya berat-berat fungsinya, kayak mount disk, perbaiki file system, atau shutdown sistem.

## 🔍 Contoh isi /sbin:

| File Biner | Fungsi |
|---|---|
| mount | Pasang file system (mount device) |
| umount | Lepas file system |
| fsck | Check dan repair file system |
| reboot | Restart sistem |
| shutdown | Matikan sistem |
| init | Inisialisasi sistem (old-style init) |
| ifconfig | Konfigurasi IP address (legacy) |
| ip | Tool modern buat konfigurasi IP/network |

| | |
|---|---|
| iptables | Firewall configurator (rule jaringan) |

❗ Biner-biner ini nggak semua user boleh akses — biasanya lo harus pakai sudo atau login sebagai root.

---

## 🧱 3. /usr → User System Resources

📌 Fungsi: Rumah besar buat program-program, libraries, dokumentasi, dll yang tidak kritis untuk booting tapi dibutuhkan setelah sistem jalan.

### 🔍 Struktur dalam /usr:

| Subdirektori | Fungsi |
|---|---|
| /usr/bin | Program executable user biasa (misal: firefox, nano, python3) |
| /usr/sbin | Program executable buat admin (misal: apachectl, sshd, usermod) |
| /usr/lib | Library file buat support program di /usr/bin & /usr/sbin |
| /usr/share | Data statis: icon, locale, man pages, dll |
| /usr/local | Aplikasi/program custom hasil install manual (misal: compile dari source) |

---

## 🧱 4. /usr/bin → User-Level Programs (Non-Kritis)

📌 Fungsi: Berisi executable file buat user biasa. Ini kayak "Program Files" Windows tapi isinya tool CLI/GUI.

### 🔍 Contoh isi /usr/bin:

| File/Tool | Fungsi |
|---|---|
| python3 | Interpreter Python |
| nano, vim | Teks editor CLI |
| firefox | Web browser |
| gcc | Compiler C/C++ |

| | |
|---|---|
| wget, curl | Download file dari internet via CLI |
| ping, traceroute | Tool jaringan |

❗ Ini isinya tool yang umum tapi tidak krusial saat boot, makanya disimpan di /usr/bin bukan /bin.

---

## 🧱 5. /usr/sbin → Admin-Level Tools (Non-Kritis)

📌 Fungsi: Berisi executable buat root/admin tapi tidak krusial saat boot.

## 🔍 Contoh isi /usr/sbin:

| File | Fungsi |
|---|---|
| sshd | SSH daemon |
| apachectl | Controller buat Apache web server |
| usermod, groupmod | Ubah data user atau grup |
| adduser, deluser | Nambah/hapus user |
| dhclient | DHCP client tool |

❗ Biasanya dipakai pas sistem udah nyala sempurna, bukan pas boot awal.

---

## 🧱 6. /lib dan /lib64 → Essential Shared Libraries

📌 Fungsi: Berisi file library bersama (.so = shared object) yang dibutuhin sama program di /bin dan /sbin.

## 🔍 Penjelasan:

- /lib → Buat sistem 32-bit
- /lib64 → Buat sistem 64-bit

## 🔍 Contoh isi:

| File | Fungsi |
| --- | --- |
| ld-linux-x86-64.so.2 | Loader utama buat library 64-bit (critical banget buat eksekusi) |
| libc.so.6 | Standar C library (dipakai hampir semua program) |
| libm.so.6 | Math library |
| libpthread.so.0 | Threading support buat program |
| libdl.so.2 | Dynamic linking support |

❗ Sama kayak .dll di Windows — kalau ini korup/hilang, program CLI bisa gagal jalan total.

---

## 📌 Summary Diagram

/bin → ls, cp, mv, rm, cat (Essential command buat semua user)

/sbin → mount, fsck, shutdown (Command buat admin)

/usr/bin → firefox, nano, curl (Program user biasa)

/usr/sbin → sshd, apachectl (Program admin non-kritis)

/lib → Library 32-bit

/lib64 → Library 64-bit

/usr → Folder besar isinya bin, sbin, lib, share, local, dll

---

Kalau lo mau gua buatin simulasi real dengan ls -l atau struktur tree directory dari masing-masing itu, tinggal bilang aja lanjut ke simulasi real bro!, gua siap drop langsung! 🔥

Oke cuuyy gassss lanjut! 🔥 Kali ini kita bedah habis subdirektori + isi penting + maksud dan tujuan dari direktori Linux yang lo sebutin berikut ini:

/usr/lib, /boot, /etc, /home, /root, /dev, /media

Masih pakai gaya semi santai anak Jaksel biar nyantol tapi tetep teknikal 👇

---

# 🧱 1. /usr/lib → Library buat Program di /usr/bin dan /usr/sbin

📌 Fungsi:
Tempat nyimpen library pendukung (.so file) buat aplikasi umum, biasanya yang ada di /usr/bin dan /usr/sbin.

## 🔍 Contoh isinya:

| File/Subdir | Fungsi |
|---|---|
| libpython3.11.so | Library runtime Python 3.11 |
| systemd/ | Komponen bawaan systemd |
| firefox/ | Library atau plugin yang dipakai sama Firefox |
| libnss_dns.so.2 | Library buat resolve DNS (nama domain → IP) |

📦 Kalau ini korup, aplikasi GUI/CLI modern bisa gagal launch.

---

# 🧱 2. /boot → Tempat File Booting Kernel & Bootloader

📌 Fungsi:
Folder khusus buat simpen semua file penting saat sistem booting, termasuk kernel, grub, initrd.

## 🔍 Contoh isinya:

| File/Subdir | Fungsi |
|---|---|
| vmlinuz- | Kernel Linux dalam format compressed |
| initramfs-.img | Initial RAM Disk buat support early boot (driver, fs, dll) |
| grub/ | Folder konfigurasi GRUB Bootloader |
| config- | Konfigurasi kernel |
| System.map- | Peta simbol buat debug kernel |

💣 Kalau folder ini rusak, sistem gak akan bisa boot!

## 🧱 3. /etc → Semua File Konfigurasi Sistem

📌 Fungsi:
Tempat semua file config sistem global disimpen. Termasuk buat network, user, service, booting, dll.

## 🔍 Contoh isinya:

| File/Subdir | Fungsi |
|---|---|
| passwd, shadow | Data user & password |
| hostname | Nama host komputer |
| resolv.conf | Config DNS resolver |
| ssh/sshd_config | Konfigurasi SSH |
| fstab | File system yang dimount otomatis saat boot |
| network/ | Konfigurasi interface jaringan |
| systemd/ | Service systemd config |
| yum.repos.d/ | Tempat repo buat CentOS/RHEL |

💡 Semua config server/aplikasi umumnya disimpan di sini.

---

## 🧱 4. /home → Tempat Folder User Biasa

📌 Fungsi:
Setiap user biasa punya "rumah" sendiri, isinya semua file user dia (document, download, setting, dll)

## 🔍 Contoh strukturnya:

/home/

├── centos/

│ ├── Documents/

│ ├── Downloads/

```
│   ├── .bashrc
│   └── .config/
└── adji/
    ├── Desktop/
    ├── Pictures/
    ├── .ssh/
```

💡 Semua file personal user ada di sini. Wajib backup sebelum resize atau oprek!

---

## 🧱 5. /root → Home-nya User root

📌 Fungsi:
Bukan /home/root, tapi langsung /root. Di sini semua file pribadi atau config khusus user root disimpan.

## 🔍 Contoh isinya:

| File/Subdir | Fungsi |
|---|---|
| .bashrc | Konfigurasi bash buat root |
| .ssh/ | Key SSH si root |
| scripts/ | Bisa aja root nyimpen script atau log hasil debug sistem |

💡 Cuma bisa diakses root user. Jadi semacam "rumah VIP" 😎.

---

## 🧱 6. /dev → Semua File Device

📌 Fungsi:
File di sini sebenernya mewakili hardware device (harddisk, cdrom, usb, ram, dll) — ini file khusus device node.

## 🔍 Contoh isinya:

| File | Fungsi |
|---|---|
| /dev/sda | Hard disk utama (entire disk) |
| /dev/sda1 | Partisi 1 dari disk /dev/sda |
| /dev/sr0 | CD/DVD ROM drive |
| /dev/null | Blackhole, buang output |
| /dev/zero | Generator nol |
| /dev/random | Generator random bytes (buat crypto) |
| /dev/tty | Terminal aktif sekarang |
| /dev/loop0 | Virtual disk dari file .iso, .img |

💡 Akses langsung hardware via CLI? Ya dari sini!

---

## 🧱 7. /media → Mount Point Otomatis Buat USB/CD

📌 Fungsi:
Kalau lo colok USB atau DVD, biasanya akan otomatis dimount di sini. Bisa per-user juga (/run/media/).

### 🔍 Contoh strukturnya saat USB colok:

/media/

└── centos/

└── USB_DISK_32GB/

├── file1.txt

└── tugas.docx

💡 Biasanya file explorer GUI kayak Nautilus otomatis mount disk di sini.

---

## 💥 Penutup Summary

| Direktori | Fungsi Utama |
|---|---|
| /usr/lib | Library support program user (di /usr/bin dan /usr/sbin) |
| /boot | File booting: kernel, grub, initrd |
| /etc | Konfigurasi sistem (user, ssh, fstab, DNS, network, service, dll) |
| /home | Tempat folder user biasa |
| /root | Home-nya user root |
| /dev | File device node untuk akses hardware |
| /media | Tempat mounting otomatis USB/DVD |

---

Kalau lo pengen lanjut eksplor subfolder daleman dari /etc (misal: systemd, network, ssh, dll), tinggal gas aja cuy bilang "lanjut ke /etc" 👊🔥

Gas terus cuyy, kita bedah tuntas lagi direktori bawaan Linux sisanya yang lo sebutin itu! 😎🔥
Masih pake gaya ringan biar makin gampang nempel di otak, tapi tetep dalem sampe akar-akarnya.

---

## 🧱 1. /run → Data Runtime Saat Sistem Aktif

📌 Fungsi:
Direktori ini dibuat ulang setiap boot. Isinya biasanya file socket, PID, lock, dan info runtime service.

## 🔍 Contoh isinya:

| File/Subdir | Fungsi |
|---|---|
| /run/sshd.pid | File penanda service sshd jalan (nyimpen PID) |
| /run/user/1000/ | Session data user UID 1000 (biasanya user biasa) |
| /run/lock/ | File lock buat mencegah proses jalan ganda |
| /run/dbus/ | File socket buat komunikasi antar proses pakai D-Bus |

💡 Semua data di sini akan hilang pas reboot.

## 🧱 2. /tmp → Tempat File Sementara

📌 Fungsi:
Folder buat simpen file sementara aplikasi atau user. Umumnya di-clear otomatis saat reboot.

## 🔍 Contoh isinya:

| File/Subdir | Fungsi |
|---|---|
| /tmp/tempfile123 | File sementara dari script atau software |
| /tmp/.X11-unix/ | Socket sementara buat server X11 (GUI) |
| /tmp/systemd-private* | Temporary sandbox dari service yang jalan via systemd |

💡 Jangan simpan file penting di sini, auto ilang tiap restart!

---

## 🧱 3. /var → Data yang Sering Berubah

📌 Fungsi:
/var = “variable”. Berisi data yang selalu berubah, seperti log, mail, cache, db spool, dll.

## 🔍 Contoh subfolder penting:

| Subdir | Fungsi |
|---|---|
| /var/log/ | Semua log sistem (syslog, secure, journal, dmesg) |
| /var/mail/ | Inbox user (kalau pake mail server lokal) |
| /var/cache/ | Cache aplikasi (dnf, apt, systemd, dll) |
| /var/spool/ | Spool data buat printer, mail, dll |
| /var/lib/ | Data persist service (mysql, systemd, etc) |
| /var/run/ | Link ke /run (buat backward compatibility) |

💡 Ini folder yang paling aktif ditulis sama sistem.

---

## 🧱 4. /opt → Tempat Aplikasi Tambahan

📌 Fungsi:
Di sinilah tempat instalasi software tambahan (yang bukan dari package manager resmi), contohnya Chrome, VirtualBox, VMware, dll.

## 🔍 Contoh isinya:

| | |
|---|---|
| Folder/File | Fungsi |
| /opt/google/chrome/ | Semua file biner & library Google Chrome |
| /opt/VirtualBox/ | File aplikasi Oracle VirtualBox |
| /opt/myapp/ | Folder aplikasi buatan lo sendiri |

💡 Umumnya dipake pas lo install aplikasi lewat .run atau .sh.

---

## 🧱 5. /proc → Virtual File Tentang Sistem & Kernel

📌 Fungsi:
Bukan folder biasa. Ini adalah virtual filesystem yang nampilin info runtime dari kernel dan proses.

## 🔍 Contoh isinya:

| | |
|---|---|
| File/Subdir | Fungsi |
| /proc/cpuinfo | Info tentang CPU (jumlah core, vendor, dll) |
| /proc/meminfo | Info RAM |
| /proc/uptime | Lama waktu sistem nyala |
| /proc// | Info proses spesifik (contoh: /proc/1234/) |
| /proc/version | Info versi kernel |

💡 Lo bisa akses info tanpa pake command top atau htop cuma via cat.

---

## 🧱 6. /sys → Virtual File Tentang Kernel & Device

📌 Fungsi:
Mirip /proc, tapi lebih fokus ke struktur kernel & hardware device.

## 🔍 Contoh isinya:

| | |
|---|---|
| Path | Fungsi |
| /sys/class/net/ | Info semua interface jaringan (eth0, lo, wlan0, dst) |
| /sys/block/ | Info disk (sda, sdb, dll) |
| /sys/devices/ | Info perangkat fisik (USB, CPU, RAM, dsb) |

💡 Semua device hardware yang dikenali kernel bakal punya representation di sini.

---

## 🧱 7. /srv → Tempat Data Buat Layanan Server

📌 Fungsi:
Kalau lo install web server, FTP server, dsb, data public-nya bisa ditaruh di sini.

## 🔍 Contoh isinya:

| | |
|---|---|
| Path | Fungsi |
| /srv/www/ | Bisa jadi tempat data buat web server kayak Apache/Nginx |
| /srv/ftp/ | Folder file public buat FTP server |

💡 Sering dipake di server production atau enterprise environment.

---

## 🧱 8. /lost+found → Folder Recovery File System

📌 Fungsi:
Otomatis dibuat saat format file system (ext4, dll). Isinya file yang berhasil direcover pas fsck (perintah check disk).

## 🔍 Contoh isinya:

| File | Fungsi |
|---|---|
| #1234567 | File yang hilang (tanpa nama) tapi berhasil diselamatkan oleh fsck |

💡 Ini cuma muncul di root file system (/) atau partisi berformat ext2/ext3/ext4.

---

## 💥 FINAL SUMMARY

| Direktori | Fungsi Utama |
|---|---|
| /run | Data runtime sementara |
| /tmp | File sementara auto-delete setelah reboot |
| /var | Data dinamis sistem (log, mail, cache, spool) |
| /opt | Instalasi aplikasi tambahan non-default |
| /proc | Info kernel dan proses aktif dalam bentuk file virtual |
| /sys | Info tentang device dan hardware yang dikenali kernel |
| /srv | Data milik layanan server (web, ftp, dll) |
| /lost+found | Tempat file yang direcover pas fsck |

---

Kalau lo pengen coba eksplor tiap folder ini langsung di VM pake command:

ls -lh /run /tmp /var /opt /proc /sys /srv /lost+found

atau untuk lihat secara rekursif dan dalem:

tree -L 2 /var

Kalau mau lanjut bahas subfolder dari /var/log atau /etc/systemd atau /proc, tinggal gas aja cuy bilang “next” 😎🔥

Yap! Dari log terakhir yang lo kasih, proses luksFormat dan cryptsetup open BERHASIL total. 😎🔥

Mari kita breakdown dan kasih penjelasan final:

---

# 🔐 Tahapan dan Bukti Keberhasilan:

## 🔸 1. luksFormat berhasil

Baris ini:

Enter passphrase for /dev/sdb:

Verify passphrase:

Dan tidak muncul error lagi (beda dengan sebelumnya yang error karena password kurang dari 8 karakter). Ini tandanya password-nya sesuai aturan (minimal 8 karakter) dan device berhasil diformat sebagai LUKS.

---

## 🔸 2. cryptsetup open berhasil

cryptsetup open /dev/sdb luks-adji-hardening-ecrypt-disk

Enter passphrase for /dev/sdb:

➡️ Kamu berhasil masukin password dengan benar dan device berhasil dibuka.

---

## 🔸 3. lsblk menunjukkan hasilnya:

└─luks-adji-hardening-ecrypt-disk 253:4 0 15G 0 crypt

✅ Ini adalah device mapper hasil dekripsi LUKS.
Artinya LUKS-nya sudah aktif dan bisa langsung kamu format dan pakai.

---

# ✅ Sekarang tinggal lanjut ke langkah berikutnya:

## 🔸 Format filesystem:

mkfs.ext4 /dev/mapper/luks-adji-hardening-ecrypt-disk

## 🔸 Mount:

mkdir /mnt/encrypted_disk

mount /dev/mapper/luks-adji-hardening-ecrypt-disk /mnt/encrypted_disk

## 🔸 Cek hasil:

df -h

lsblk

---

## 🎉 Kesimpulan:

- Sebelumnya gagal karena password kurang dari 8 karakter.
- Sekarang berhasil karena password memenuhi syarat.
- LUKS device berhasil dibuat, dibuka, dan siap diformat/digunakan.

Kalau lo butuh auto-mount saat booting via /etc/crypttab dan /etc/fstab, tinggal bilang aja, nanti gue bantuin buatkan juga.

Mantap nih pertanyaannya cuyy! 🔥 Lo udah makin dalem paham soal Linux & storage. Sekarang kita bedah perbedaan mount, umount, dan eject biar gak bingung lagi pas ngurus harddisk, SSD, atau flashdisk.

---

## ⚙️ 1. mount

📌 Fungsi:
mount itu proses "nempelin" (attach) file system dari device (kayak flashdisk, harddisk, partisi) ke direktori tertentu di sistem lo, supaya bisa diakses.

## 🧠 Analoginya:

Bayangin lo colok flashdisk ⇒ tapi belum lo buka di file manager ⇒ maka belum "dipasang". Nah, mount = lo pasang ke sistem.

## 🍼 Contoh:

sudo mount /dev/sdb1 /mnt/flashku

📌 Artinya: pasang partisi /dev/sdb1 ke direktori /mnt/flashku
➡ Baru setelah itu file-filenya bisa diakses dari situ.

---

## 🔌 2. umount

📌 Fungsi:
umount itu proses "lepasin" (detach) file system dari sistem. Jadi kayak bilang ke sistem:

“Yo, gue udah selesai pake device ini, stop akses ya biar aman dilepas.”

## 🔥 Kenapa penting?

Kalau lo cabut device tanpa umount, bisa bikin:

- File korup
- Data hilang
- File system rusak

## 🧪 Contoh:

sudo umount /mnt/flashku

📌 Artinya: lepas flashdisk yang tadi dipasang ke /mnt/flashku

---

## 💿 3. eject

📌 Fungsi:
eject = gabungan dari umount + sinyal “boleh cabut” ke perangkat. Biasanya dipake buat CD/DVD-ROM, USB, HDD eksternal, dll.

💡 Kalau device-nya mendukung, perintah eject juga bisa motong power ke port USB-nya (khusus perangkat USB tertentu aja).

## 🧪 Contoh:

sudo eject /dev/sdb

📌 Artinya: lepas device /dev/sdb dan kasih sinyal aman untuk dicabut.

---

## 🔍 Bedanya Secara Tabel

| Aksi | Tujuan Utama | Bisa Akses Data? | Boleh Dicabut? | Power Off Port? |
|---|---|---|---|---|
| mount | Pasang file system ke sistem | ✅ Ya | ❌ Belum | ❌ Tidak |
| umount | Lepasin file system | ❌ Tidak bisa | ✅ Ya | ❌ Tidak |
| eject | Umount + sinyal cabut (power-off USB) | ❌ Tidak bisa | ✅ Ya | ✅ Kadang bisa |

---

## 💡 Tips Aman Nge-Flashdisk di Linux

Kalau lo colok flashdisk /dev/sdb1, cara cabut aman:

sudo umount /dev/sdb1

Atau langsung:

sudo eject /dev/sdb

---

Kalau mau, gua bisa bantuin bikin script otomatis buat mount/umount/eject flashdisk lo biar tinggal klik 😎

Tinggal bilang aja: “Buatkan script mount flashdisk otomatis bro!” 💻

Wah pertanyaan lo makin tajem cuy! 🔪🔥
Oke gua jelasin nih si /mnt/flashku itu maksudnya apa biar lu paham dari akar sampe ke rantingnya.

---

## 🎯 /mnt/flashku Itu Apa?

📌 **/mnt/flashku** = folder tempat file system device ditampilin di sistem Linux setelah lo mount.

### Gampangnya:

Lo kan punya flashdisk (/dev/sdb1)
Tapi Linux butuh tempat di sistem lo buat nampilin isi flashdisk itu
👉 Nah tempatnya itu bisa lo tentuin sendiri, contohnya /mnt/flashku.

## 🔍 Breakdown Konsepnya:

| | |
|---|---|
| Istilah | Maksud |
| /dev/sdb1 | Device fisik partisi flashdisk lo |
| /mnt/flashku | Folder (di sistem lo) untuk nampilin isi flashdisk setelah di-mount |

---

## 🧠 Analoginya Gampang

Bayangin:

- Lo punya isi koper (isi flashdisk)
- Tapi lo gak bisa buka koper di sembarang tempat
- Lo butuh meja (mount point) buat buka koper dan ngambil isinya

Jadi:

sudo mount /dev/sdb1 /mnt/flashku

Artinya:

"Gue pengen buka flashdisk (/dev/sdb1) dan nampilin isinya di meja gue (/mnt/flashku)."

---

## 📁 Kenapa Pakai /mnt?

- Direktori /mnt itu secara konvensi Linux dipakai buat mounting sementara (temporary mount).

Boleh aja lo bikin folder lain misalnya:

sudo mkdir /media/usbku

sudo mount /dev/sdb1 /media/usbku

- 
- Tapi /mnt adalah default tempat aman buat ngetes atau pasang manual.

## 💥 Bonus: Cara Cek Isinya Setelah Mount

ls /mnt/flashku

Kalau berhasil, lo bakal lihat isi flashdisk di situ.

---

## ✨ Kesimpulan Simpel

| | |
|---|---|
| Perintah | Artinya |
| sudo mount /dev/sdb1 /mnt/flashku | Pasang flashdisk lo ke folder /mnt/flashku |
| sudo umount /mnt/flashku | Lepas flashdisk dari folder itu |
| ls /mnt/flashku | Lihat isi flashdisk yang udah ter-mount |

---

Kalau lo mau gua buatin script otomatis mount flashdisk ke folder khusus, tinggal bilang:

"Bro buatin script mount flashdisk ke folder otomatis dong" 💻🧠

Yoi cuy, gua contohin lengkap dan realistik banget kayak lo lagi colok flashdisk di VirtualBox atau real hardware!
Lo bakal lihat mulai dari sebelum mount, pas mount, lihat isinya, sampai umount.

---

## 📦 Skenario Simulasi

- Lo punya flashdisk (anggap udah ke-detect di /dev/sdb1)
- Lo mau pasang (mount) ke folder /mnt/flashku

---

## 🔧 1. Cek Dulu Device-nya

lsblk

📩 Contoh hasil:

NAME  MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS

sda  8:0  0  64G  0 disk

├─sda1  8:1  0  1G  0 part /boot

└─sda2  8:2  0  63G  0 part /

sdb  8:16  1  16G  0 disk

└─sdb1  8:17  1  16G  0 part

🧠 sdb1 = partisi flashdisk
➡️ Belum ada MOUNTPOINT = berarti belum di-mount

---

## 📁 2. Bikin Folder Tujuan Mount (mount point)

sudo mkdir -p /mnt/flashku

- -p artinya bikin folder beserta parent-nya kalau belum ada

---

## 🚀 3. Mount Flashdisk ke Folder Tadi

sudo mount /dev/sdb1 /mnt/flashku

✅ Kalau sukses, gak akan muncul pesan error.

---

## 🔍 4. Lihat Isi Flashdisk

ls /mnt/flashku

📩 Contoh hasil:

film_keren.mp4

skripsi_bang_adji.docx

foto_masa_lalu.jpeg

➡️ Artinya isi flashdisk udah bisa lo akses dari folder /mnt/flashku

## 📉 5. Umount Kalau Udah Gak Dipakai

sudo umount /mnt/flashku

📩 Kalau sukses:

- Gak ada pesan error

Lo bisa cek lagi dengan:

lsblk

- Maka bagian sdb1 udah gak ada "MOUNTPOINT"-nya lagi

---

## 🧠 BONUS: Cek Mount Status Manual

mount | grep sdb1

📩 Kalau kosong, artinya flashdisk udah ke-umount

---

## 📸 Contoh Ringkasan Output Asli:

[root@centos ~]# lsblk

sda      8:0    0   64G  0 disk

├─sda1   8:1    0    1G  0 part /boot

└─sda2   8:2    0   63G  0 part /

sdb      8:16   1   16G  0 disk

└─sdb1   8:17   1   16G  0 part

[root@centos ~]# mkdir /mnt/flashku

[root@centos ~]# mount /dev/sdb1 /mnt/flashku

[root@centos ~]# ls /mnt/flashku

film_keren.mp4  skripsi_bang_adji.docx  foto_masa_lalu.jpeg

[root@centos ~]# umount /mnt/flashku

[root@centos ~]# mount | grep sdb1

[root@centos ~]#

---

Kalau lo mau, gua bisa buatin script auto-mount setiap lo colok flashdisk, atau auto mount ke /media/usbku misalnya. Tinggal bilang aja:

"Buat script automount flashdisk dong cuy!" 💽⚙️

Nah ini dia, gua kasih penjelasan tujuan inti dari mount dan umount secara santai tapi teknikal banget:

---

# 🔧 Tujuan mount

## ✅ Tujuan utama:

"Nempelin" sistem file dari device (harddisk, SSD, flashdisk, partisi, ISO, dll) ke sistem Linux biar bisa diakses.

## 🧠 Analogi gampang:

Colok flashdisk ke PC ⇒ tapi lo belum buka foldernya ⇒ Linux belum tahu lo mau ngakses isi flashdisk itu dari mana.
📌 Nah mount = bilang ke Linux:
"Gue mau akses partisi ini lewat folder sini ya!"

---

## 📥 Contoh kenyataan:

mount /dev/sdb1 /mnt/flashku

Artinya:

Gue tempelin flashdisk /dev/sdb1 ke folder /mnt/flashku, jadi semua file di flashdisk bisa diakses dari situ.

Kalau lo gak mount, isinya gak bisa dibuka bahkan gak bisa lo cd atau ls.

---

## 🧠 Tujuan umount

## ✅ Tujuan utama:

"Ngelepas" file system dari device yang udah dipakai sistem, biar gak rusak saat dicabut/dimodif.

## ⚠️ Kenapa harus umount?

- Biar data gak korup
- File yang masih diproses diselesaikan dulu
- Cache ditulis permanen ke disk (flush)

📌 Kalau langsung cabut flashdisk tanpa umount, risiko:

- File hilang
- Data korup
- File system error

---

## 🎯 Kesimpulan

| | |
|---|---|
| Perintah | Tujuan |
| mount | Nempelin partisi/device ke folder tertentu supaya bisa diakses |
| umount | Ngelepas partisi dari sistem dengan aman supaya gak ada data yang rusak |

---

Kalau lo pakai GNOME/KDE dan colok flashdisk ⇒ itu sebenernya di-auto mount ke /run/media/namapengguna/namadevice
Terus pas klik “Eject” itu sebenernya ngejalanin umount.

---

Kalau lo mau gua buatin diagram ilustrasi mount-umount, tinggal bilang aja ya bro 💻📂

Nah mantap lanjut! 😎 Sekarang kita bahas...

---

# 🔌 Tujuan Perintah eject di Linux

## ✅ Fungsi utama:

eject itu buat mengeluarkan atau memutus akses fisik ke device, terutama:

- CD/DVD-ROM (secara harfiah tray-nya bakal keluar)
- Flashdisk atau storage lain (optional): bisa juga dipakai untuk safe removal di beberapa environment

---

## 🔧 Perbedaan dengan umount:

| Perintah | Tujuan Teknis | Perilaku Fisik | Cocok Untuk |
|---|---|---|---|
| mount | Nempelin file system | Nggak ada efek fisik | Semua storage |
| umount | Ngelepas file system | Nggak ada efek fisik | Semua storage |
| eject | Umumnya buat eject CD/DVD, tapi bisa untuk USB (di desktop) | Tray CD-ROM keluar atau device 'diputus' | CD/DVD-ROM, kadang USB |

---

## 🎯 Contoh Kasus:

### 1. Eject CD/DVD

eject /dev/sr0

📌 Tray CD-ROM akan keluar secara fisik.

### 2. Eject Flashdisk (opsional tergantung desktop)

eject /dev/sdb

📌 Di GNOME/KDE biasanya ini sama kayak klik “Safely Remove Drive”
📌 Bisa juga jadi shortcut: umount + power off port USB

## ⚠️ Catatan Penting:

- eject tidak otomatis melakukan umount (tapi di beberapa distro, bisa tergantung konfigurasi).
- Sebaiknya umount dulu, baru eject kalau device mendukung.

---

## 🧠 Gampangnya gini:

mount = pasang
umount = lepas
eject = lepas + keluarin (fisik/off secara elegan)

---

Kalau lo colok flashdisk ke laptop, terus lo klik kanan → “Eject” atau “Safely Remove”, itu di balik layar jalan:

umount /dev/sdb1 && eject /dev/sdb

---

Kalau mau gua buatin simulasi CLI-nya dari awal colok sampai eject, tinggal bilang bro 😁 🔥

💡 Pertanyaan lo keren banget cuy!
*Apakah bisa "hapus" enkripsi LUKS dan balikin disk jadi normal kayak sebelum dienkripsi?

✅ Jawaban singkatnya: YES, BISA BANGET
Tapi nggak bisa di-undo langsung — lu harus format ulang disk tersebut karena LUKS itu sistem enkripsi di level block device. Jadi, lo perlu hapus struktur LUKS-nya lalu buat filesystem baru dari nol.

---

# 🔥 Langkah-Langkah Menghapus Enkripsi LUKS dari Disk

## 🧼 1. Tutup Mapping Enkripsi Dulu

Kalau LUKS-nya masih terbuka (dalam contoh lo: luks-adji-hardening-ecrypt-disk), tutup dulu:

cryptsetup close luks-adji-hardening-ecrypt-disk

⚠️ Pastikan LV/partisi terkait tidak sedang di-mount sebelum ditutup.

---

## 🔥 2. Overwrite Struktur LUKS (Format Disk)

Misalnya device-nya /dev/sdb:

### Pilihan A: Format dengan ext4

mkfs.ext4 /dev/sdb

### Pilihan B: Zero-fill pakai dd (hancurkan data LUKS-nya total):

dd if=/dev/zero of=/dev/sdb bs=1M status=progress

Ini menghapus seluruh isi disk — termasuk LUKS header, partisi, data, segalanya.

---

## 🧪 3. Cek Apakah LUKS Sudah Hilang

cryptsetup isLuks /dev/sdb && echo "Masih LUKS" || echo "Udah bukan LUKS"

---

## 🗂️ 4. (Opsional) Buat Partisi & Filesystem Baru

fdisk /dev/sdb

# Buat partisi baru → w (write)

Lalu:

mkfs.ext4 /dev/sdb1

mkdir /mnt/mydisk

mount /dev/sdb1 /mnt/mydisk

---

## 🔐 Ringkasan

| | |
|---|---|
| Aksi | Perintah |
| Tutup LUKS | cryptsetup close |
| Hapus LUKS | mkfs.ext4 atau dd if=/dev/zero |
| Cek LUKS | cryptsetup isLuks /dev/sdX |
| Buat partisi baru | fdisk /dev/sdX |

---

## 🚨 Catatan Penting:

- Setelah di-luksFormat, nggak ada tombol undo.
- Satu-satunya cara "hapus" enkripsi adalah overwrite disk dan buat ulang filesystem.
- Semua data di dalamnya akan hilang total kecuali lo backup dulu sebelum luksFormat.

---

Kalau lo mau coba simulasi restore data terenkripsi atau gabungin LUKS sama LVM (nested encryption) gua siap bantu juga, tinggal bilang aja 😎
Mau lanjut kemana lagi coach? 🔧

Siap bro! Nih gua kasih simulasi full step-by-step hapus enkripsi LUKS dari disk dan balikin dia ke kondisi non-encrypted (normal).
Kita bakal:

1. 🔐 Encrypt disk pake LUKS
2. 🔓 Open/mount disk
3. 💣 Tutup LUKS + Hapus enkripsi (hapus LUKS header)
4. 📁 Buat partisi & filesystem baru (jadi disk normal)
5. 🔍 Verifikasi hasilnya

---

## 💡 Simulasi: Hapus Enkripsi LUKS dari Disk

Contoh disk: /dev/sdb
Mapping name: luks_test_disk

## 🧱 STEP 1 – Encrypt Disk (LUKS)

cryptsetup luksFormat /dev/sdb

Are you sure? (Type 'YES'): YES

Enter passphrase for /dev/sdb:

Verify passphrase:

---

## 🔓 STEP 2 – Buka LUKS & Buat Filesystem

cryptsetup open /dev/sdb luks_test_disk

Lalu buat filesystem di dalam mapping:

mkfs.ext4 /dev/mapper/luks_test_disk

Mount:

mkdir /mnt/luks_disk

mount /dev/mapper/luks_test_disk /mnt/luks_disk

Cek:

lsblk

Outputnya kira-kira:

sdb 8:16 0 15G 0 disk

└─luks_test_disk 253:4 0 15G 0 crypt /mnt/luks_disk

---

## ❌ STEP 3 – Hapus Enkripsi LUKS (Bikin Jadi Normal Disk)

### 🧹 A. Unmount dan Close

umount /mnt/luks_disk

cryptsetup close luks_test_disk

### 🔥 B. Overwrite LUKS Header / Format Disk Ulang

mkfs.ext4 /dev/sdb

Atau, untuk benar-benar hancurin total struktur enkripsinya:

dd if=/dev/zero of=/dev/sdb bs=1M count=100 status=progress

count=100 cukup untuk menghapus LUKS header. Bisa diperbesar untuk full-wipe.

---

## 🧪 STEP 4 – Cek Ulang

cryptsetup isLuks /dev/sdb && echo "Masih LUKS" || echo "Udah normal disk"

✅ Output-nya:

Udah normal disk

---

## 📁 STEP 5 – Buat Partisi dan Filesystem Baru

fdisk /dev/sdb

# Tekan: o (buat DOS label), n (new partisi), w (write)

mkfs.ext4 /dev/sdb1

mkdir /mnt/normal_disk

mount /dev/sdb1 /mnt/normal_disk

Cek:

lsblk

Output:

sdb 8:16 0 15G 0 disk

└─sdb1 8:17 0 15G 0 part /mnt/normal_disk

---

## 📌 Hasil Akhir

- Disk /dev/sdb udah jadi non-encrypted
- Bisa langsung dipake buat simpan data tanpa LUKS
- Tidak ada crypt mapping lagi
- Tidak perlu password saat boot/mount

---

Kalau mau gua bantu bikinin skrip otomatis buat hapus LUKS + reset disk, tinggal bilang ya 🔥
Atau lanjut ke simulasi nested LUKS + LVM kalau pengen lebih hardcore 💀

Mau lanjut ke mana, Coach? 😎

Oke coach, sekarang gua ajarin lo cara bikin partisi dari nol di Linux dari harddisk/SSD kosong — no file system, no partition table, bener-bener fresh kayak bayi baru lahir 😄

---

## 🎯 Target:

Buat partisi dari disk kosong (misal: /dev/sdb), lalu siapin buat dipake.

---

## 📦 LANGKAH-LANGKAH BUAT PARTISI DARI NOL DI LINUX

Tools yang kita pake: fdisk (bisa juga parted, tapi fdisk lebih familiar)

---

### 🧭 1. Cek dulu disk kosongnya

lsblk

atau

fdisk -l

Contoh output:

sdb      8:16   0   15G  0 disk

Kita asumsikan /dev/sdb adalah disk kosong (❌ jangan pake /dev/sda karena itu bisa OS lu!).

---

## 🛠️ 2. Masuk ke fdisk buat ngatur partisi

fdisk /dev/sdb

Lo masuk ke mode interaktif. Berikut ini opsi yang perlu lu tekan:

| Tombol | Fungsi |
| --- | --- |
| o | Buat new DOS partition table (bersihin semua struktur lama) |
| n | Buat partisi baru |
| p | Primary partition |
| (Enter) | Default partisi pertama |
| (Enter) | Start sector (default) |
| (Enter) | End sector (default = all size) |
| w | Tulis perubahan ke disk dan keluar |

---

## 💿 Contoh Interaktif:

Command (m for help): o

Created a new DOS disklabel with disk identifier 0xXXXXXXX.

Command (m for help): n

Partition type:

p primary

e extended

Select (default p): p

Partition number (1-4, default 1):

First sector (2048-..., default 2048):

Last sector, +sectors or +size{K,M,G,T,P} (default ...):

Created a new partition 1 of type 'Linux' and of size 15 GiB.

Command (m for help): w

---

## 🧹 3. Format partisi yang baru dibuat

Setelah selesai, partisi baru akan muncul sebagai /dev/sdb1

mkfs.ext4 /dev/sdb1

Kalau mau pakai xfs:

mkfs.xfs /dev/sdb1

---

## 📂 4. Mount ke folder supaya bisa dipakai

mkdir /mnt/mydisk

mount /dev/sdb1 /mnt/mydisk

Cek:

lsblk

Output:

sdb 8:16 0 15G 0 disk

└─sdb1 8:17 0 15G 0 part /mnt/mydisk

---

## 🔁 (Opsional) Auto-mount pas boot (fstab)

Tambahkan ke /etc/fstab:

echo "/dev/sdb1 /mnt/mydisk ext4 defaults 0 0" >> /etc/fstab

---

## ✅ HASIL AKHIR:

- 🧱 Disk /dev/sdb udah dipartisi jadi /dev/sdb1
- 💾 Sudah diformat pakai ext4 dan bisa dipake
- 📂 Bisa diakses lewat /mnt/mydisk
- 🧠 Siap dipakai buat simpan data, backup, atau dijadiin LVM

---

Kalau mau gua ajarin:

- 💣 Multi-partisi (root/home/swap),
- 🧬 GPT vs MBR,
- 🚀 Pakai parted untuk disk besar (>2TB),
- atau 🔐 Digabungin sama LUKS/LVM, tinggal gas aja bro 😎

Mau lanjut ke mana nih, Coach?

Wah ini pertanyaan mantap bro! 😎 Kita bahas tuntas yuk:

---

# 💥 MBR vs GPT – Si Dua Jejaka Penjaga Disk!

## 🎯 Tujuan Utama:

MBR dan GPT adalah partisi tabel (partition table) — mereka ngatur gimana harddisk/SSD lo dibagi-bagi, dibaca, dan dipake sama sistem operasi.

Kalau disk itu rumah, partition table = arsitek yang ngatur denahnya.

---

## 🆚 Bedanya Apa? Nih dia:

| Perbandingan | 👴 MBR (Master Boot Record) | 🚀 GPT (GUID Partition Table) |
|---|---|---|
| Rilis | 1983 (jaman BIOS jadul) | 2000-an (zaman modern) |
| Kapasitas Maks | 2 TB per disk | > 9 ZB (Zettabyte = 1 Milyar TB) 😱 |

| | | |
|---|---|---|
| Jumlah Partisi | Max 4 Primary (atau 3 Primary + 1 Extended) | 128 partisi langsung |
| Dukungan Boot | Hanya BIOS (legacy) | BIOS + UEFI (modern) |
| Struktur | Simpel tapi rentan (1 partisi table doang) | Redundant (backup GPT table + CRC32 check) |
| Recovery | Gampang korup, susah recovery | Lebih tahan error (ada backup-nya) |
| Disk Format | Cocok untuk disk <2TB | Harus pake kalau disk >2TB |
| Kompatibilitas | Lebih kompatibel sama OS lama (Windows XP 32-bit) | Butuh OS modern (Windows 10/11, Linux modern) |

---

## 📌 Kesimpulan Singkat

| | |
|---|---|
| Kalau lo... | Pakai... |
| Disk < 2TB & butuh support OS lama | MBR |
| Disk modern, >2TB, performa tinggi, stabil, future-proof | GPT ✅ |

---

## 🧠 Cara Kerja Singkat

### 👴 MBR

- MBR tersimpan di 512 byte pertama dari disk (sector 0).
- Isi:
- Bootloader (446 byte)
- Partition Table (64 byte)
- Magic Number (2 byte)
- Maksimal 4 partisi karena slot partisinya cuma ada 4.

### 🚀 GPT

- Ada di bagian awal dan akhir disk.
- Tiap partisi punya UUID (ID unik).

- Ada CRC checksum buat validasi data.
- Bisa bikin ratusan partisi dan dukung disk raksasa.

---

## 💡 Contoh Implementasi di Linux

### Cek partisi table disk:

parted /dev/sdb print

Kalau MBR:

Partition Table: msdos

Kalau GPT:

Partition Table: gpt

### Ganti ke GPT:

parted /dev/sdb

(parted) mklabel gpt

Kalau mau pake MBR:

(parted) mklabel msdos

---

## ⚠️ NOTE PENTING

- GPT WAJIB pake UEFI BIOS, sedangkan MBR cuma jalan di BIOS legacy.
- OS lama (kayak Windows XP) gak bisa baca GPT.
- GPT jauh lebih aman & future-ready.

---

Mau gua bikinin simulasi install Linux pake disk GPT vs MBR? Atau lanjut ke topik lain misalnya UEFI vs BIOS, dual-boot, atau disk cloning? 😏

Woke Coachh! 🚀 Sekarang kita bahas topik lanjutan:

---

# 🔧 Jenis-Jenis Partisi di Linux dan Windows

Alias si pembagi lahan di harddisk/SSD lo 💽

---

## 🧱 1. Primary Partition (Partisi Primer)

### 🧠 Pengertian:

Partisi utama yang langsung bisa dipake buat booting OS atau nyimpen data.

### 📦 Ciri-ciri:

- Bisa bikin maksimal 4 Primary Partition di MBR.
- GPT bisa punya sampe 128 partisi tanpa batasan primary/extended.
- Bisa dijadiin tempat OS.

### 📌 Contoh:

/dev/sda1 → partisi primary buat /boot

/dev/sda2 → partisi primary buat root (/)

---

## 📦 2. Extended Partition (Partisi Perluasan)

### 🧠 Pengertian:

Partisi khusus buat nampung partisi lain (logical), karena MBR cuma boleh punya 4 partisi total.

### 📦 Ciri-ciri:

- Cuma boleh ada 1 Extended Partition per disk.
- Dibikin kalo lo butuh lebih dari 4 partisi di MBR.
- Gak bisa langsung dipakai buat nyimpen data.

### 📌 Contoh:

/dev/sda3 → extended (nanti isinya logical partisi)

---

## 🧩 3. Logical Partition (Partisi Logikal)

### 🧠 Pengertian:

Partisi yang berada di dalam Extended Partition. Ini solusi untuk ngelampaui limit 4 partisi di MBR.

### 📦 Ciri-ciri:

- Gak bisa booting OS secara langsung (biasanya), tapi bisa dipakai buat nyimpen /home, /var, dll.
- Nomornya dimulai dari /dev/sda5, karena sda1-sda4 buat primary/extended.

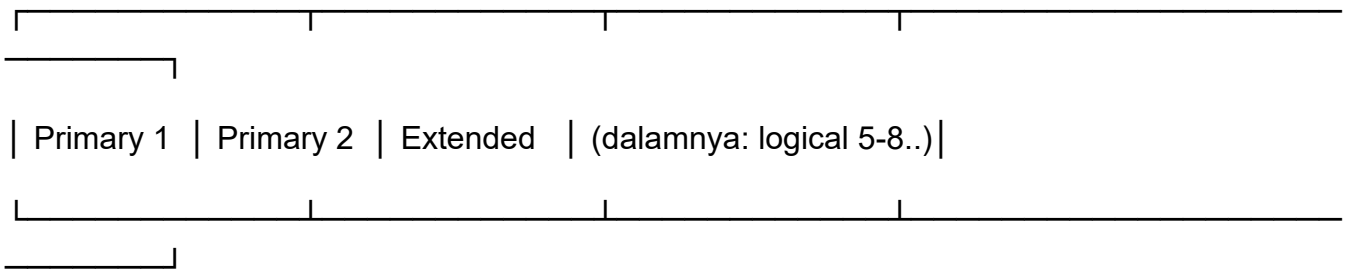
### 📌 Contoh:

/dev/sda5 → logical di dalam extended (buat /home)

---

## 💡 Gimana Diagramnya di MBR?

MBR:




---

## 🚀 GPT Nggak Ribet Kayak MBR

Kalau pake GPT, gak ada konsep extended/logical. Semua partisi dianggap setara (equal / primary style).

Jadi bisa langsung:

/dev/sda1 → EFI

/dev/sda2 → /

/dev/sda3 → /home

...

/dev/sda10 → /backup

---

## 🔧 Contoh Praktik Partisi (MBR)

fdisk /dev/sdb

1. Tekan n → new
2. Pilih p → primary
3. Ulang sampe partisi ke-3
4. Untuk ke-4, pilih e → extended
5. Lalu dalamnya bikin n → logical
6. Format: mkfs.ext4 /dev/sdb5
7. Mount: mount /dev/sdb5 /mnt/data

---

## ✅ Kesimpulan

| Tipe Partisi | Bisa Boot | Maks Jumlah (MBR) | Tujuan |
|---|---|---|---|
| Primary | ✅ | 4 total (MBR) | OS, data |
| Extended | ❌ | 1 aja | Nampung logical |
| Logical | ❌/⚠️ | Bebas (5 ke atas) | Data / struktur tambahan |
| GPT Partition | ✅ | Sampai 128++ | Semua (tanpa batasan) |

---

Kalau mau gua bikinin simulasi:

- Buat MBR vs GPT partisi
- Buat dual boot / multiboot pake partisi berbeda
- LVM + Partisi
  Tinggal gas ya bro! 😎

Siapp broo! 🤓 Kita langsung gas ke contoh praktik partisi dari awal di Linux, pake tool fdisk, yang paling umum dan ada di hampir semua distro. Simulasi ini anggap kita mau ngebagi harddisk kosong /dev/sdb ya.

---

# 🧪 Simulasi: Membuat Partition di Linux dengan fdisk

## ✅ Langkah Awal

lsblk

Pastikan disk kosongnya terdeteksi, misal /dev/sdb:

NAME  MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS

sda  8:0  0  64G  0 disk

├─sda1  8:1  0  1G  0 part /boot

└─sda2  8:2  0  63G  0 part

sdb  8:16  0  15G  0 disk  ← Disk kosong

---

## 🛠️ Langkah 1: Buka Disk pake fdisk

fdisk /dev/sdb

---

## 📚 Menu fdisk Penting

| | |
|---|---|
| Perintah | Fungsi |
| n | Buat partisi baru |
| d | Hapus partisi |
| p | Tampilkan tabel partisi |
| t | Ubah tipe partisi |
| w | Tulis & simpan |
| q | Keluar tanpa simpan |

## 💿 Buat 2 Primary Partition

### ➤ Primary #1: 2GB

Command (m for help): n

Partition type: p

Partition number: 1

First sector: (ENTER)

Last sector: +2G

### ➤ Primary #2: 5GB

Command (m for help): n

Partition number: 2

First sector: (ENTER)

Last sector: +5G

---

## 💿 Buat Extended Partition

Command (m for help): n

Partition type: e

Partition number: 3

First sector: (ENTER)

Last sector: (ENTER)

---

## 💿 Buat Logical Partition dalam Extended

### ➤ Logical #1: 3GB

Command (m for help): n

Partition number: (otomatis jadi 5)

First sector: (ENTER)

Last sector: +3G

---

## ✅ Tulis dan Simpan

Command (m for help): w

---

## 🔄 Format Partisi

Contoh format ke ext4:

mkfs.ext4 /dev/sdb1

mkfs.ext4 /dev/sdb2

mkfs.ext4 /dev/sdb5

---

## 📦 Mount Partisi

mkdir /mnt/data1 /mnt/data2 /mnt/data3

mount /dev/sdb1 /mnt/data1

mount /dev/sdb2 /mnt/data2

mount /dev/sdb5 /mnt/data3

---

## 📊 Cek Hasil

lsblk

Output-nya bakal muncul gini:

```
sdb      8:16   0   15G  0 disk
├─sdb1   8:17   0    2G  0 part  /mnt/data1
├─sdb2   8:18   0    5G  0 part  /mnt/data2
├─sdb3   8:19   0    1  0 part  (extended)
└─sdb5   8:20   0    3G  0 part  /mnt/data3
```

---

## 🔥 Bonus: Buat Partisi GPT

Kalau pake disk modern dan lo mau GPT:

```
parted /dev/sdb

(parted) mklabel gpt

(parted) mkpart primary ext4 0% 30%

(parted) mkpart primary ext4 30% 60%

(parted) mkpart primary ext4 60% 100%

(parted) quit
```

Lanjut mkfs.ext4 dan mount seperti biasa.

---

Kalau mau lanjut simulasi ke:

- Auto-mount via /etc/fstab
- Pasang LUKS encryption ke partisi
- Gabung ke LVM
- Resize & delete

Kasi kode: "LANJUT PARTISI LEVEL 2" 😎

Siapp bro! 🔥 Kita gaspol ke contoh lengkap pembuatan partisi GPT dari nol sampai bisa dipake, cocok buat disk modern/SSD NVMe yang lo punya.

---

# ⚙️ Simulasi Lengkap: Partisi GPT dengan parted + Format + Mount

## 📀 1. Cek Disk Kosong

lsblk

Misalnya target kita: /dev/sdb
Kondisinya masih kosong dan belum ada partisi.

---

## 💡 2. Mulai Pake parted (Bukan fdisk, karena ini GPT)

parted /dev/sdb

Lalu masuk mode interaktif parted, kita bikin label GPT dulu:

(parted) mklabel gpt

💬 Ini ngapus semua isi disk dan ganti jadi GPT!

---

## 🧱 3. Buat 3 Partisi Primary

### ➤ Partisi 1: 0%–30% (misal 5GB)

(parted) mkpart primary ext4 0% 30%

### ➤ Partisi 2: 30%–60% (misal 5GB)

(parted) mkpart primary ext4 30% 60%

### ➤ Partisi 3: 60%–100% (sisa disk)

(parted) mkpart primary ext4 60% 100%

Akhiri sesi:

(parted) quit

---

## 🧼 4. Format Semua Partisi ke ext4

mkfs.ext4 /dev/sdb1

mkfs.ext4 /dev/sdb2

mkfs.ext4 /dev/sdb3

---

## 📂 5. Mount Partisinya ke Folder

mkdir -p /mnt/gpt1 /mnt/gpt2 /mnt/gpt3

mount /dev/sdb1 /mnt/gpt1

mount /dev/sdb2 /mnt/gpt2

mount /dev/sdb3 /mnt/gpt3

---

## ✅ 6. Cek Hasilnya

lsblk -f

Output contoh:

NAME FSTYPE LABEL UUID MOUNTPOINT

sdb

├─sdb1 ext4 e1d4e2f4-1234-4c42-b123-aaa111bbb222 /mnt/gpt1

├─sdb2 ext4 f2c3d3e4-5678-4c42-b321-bbb222ccc333 /mnt/gpt2

└─sdb3 ext4 a3e4f5e6-7890-4c42-c456-ccc333ddd444 /mnt/gpt3

---

## 🚀 Tambahan: Auto-Mount ke /etc/fstab

Contoh baris /etc/fstab:

UUID=e1d4e2f4-1234-4c42-b123-aaa111bbb222 /mnt/gpt1 ext4 defaults 0 2

UUID=f2c3d3e4-5678-4c42-b321-bbb222ccc333 /mnt/gpt2 ext4 defaults 0 2

UUID=a3e4f5e6-7890-4c42-c456-ccc333ddd444 /mnt/gpt3 ext4 defaults 0 2

🔍 UUID bisa dilihat dengan:

blkid

---

Kalau mau lanjut ke:

- GPT + LUKS
- GPT + LVM
- Resize partisi GPT
- GPT + Bootable disk

Ketik aja: “LANJUT GPT LEVEL 2” 😎

🔥 Great question bro, ini penting banget sebelum bikin partisi, apalagi kalo udah masuk ke MBR vs GPT!

---

# 📜 Aturan Main saat Membuat Partisi di MBR & GPT

## ⚠️ MBR (Master Boot Record)

MBR itu sistem partisi lawas, dan punya batasan ketat:

| | |
|---|---|
| Aturan | Penjelasan |
| ❌ Maks 4 partisi primary | Cuma bisa bikin 4 partisi utama (primary) doang |
| ✅ Extended + Logical | Kalau butuh lebih dari 4 partisi, harus bikin 1 extended dan sisanya jadi logical di dalamnya |
| ❌ Kapasitas disk terbatas | Cuma bisa manage disk sampai 2TB |
| ✅ Bootable support jadul | Masih cocok buat sistem BIOS lama (non-UEFI) |

🧠 Contoh Struktur MBR (max 6 partisi):

- /dev/sda1 → primary
- /dev/sda2 → primary
- /dev/sda3 → primary
- /dev/sda4 → extended
- /dev/sda5 → logical
- /dev/sda6 → logical

---

## ✅ GPT (GUID Partition Table)

GPT itu sistem modern, dan jauh lebih fleksibel:

| Aturan | Penjelasan |
|---|---|
| ✅ Maks 128 partisi | Gak ada drama extended/logical, langsung bisa bikin hingga 128 partisi primary |
| ✅ Ukuran disk gede | Support disk sampe 9.4 Zettabyte (iya ZB, bukan TB bro 😆) |
| ✅ UEFI Boot | Wajib dipakai buat sistem UEFI modern |
| ✅ CRC protection | Lebih aman: GPT nyimpen backup table & checksum CRC di awal dan akhir disk |

🧠 Contoh Struktur GPT:

- /dev/sdb1 → primary
- /dev/sdb2 → primary
- /dev/sdb3 → primary
- ...
- /dev/sdb128 → primary 😱

---

## 🔍 Aturan Umum (MBR & GPT)

- Gunakan MBR hanya jika:
- Sistem lama (BIOS only)
- Disk < 2 TB
- Mau dual boot sama OS lawas (Windows XP misalnya)

- Gunakan GPT jika:
- Disk > 2 TB
- Sistem udah UEFI
- Butuh banyak partisi
- Mau lebih future-proof

---

## 🚧 Error yang Sering Muncul

| | | |
|---|---|---|
| Error | Penyebab | Solusi |
| "Cannot create more than 4 primary partitions" | MBR limit | Gunakan extended/logical atau pindah ke GPT |
| "Boot failed after install" | GPT di BIOS-only sistem | Convert GPT → MBR atau enable UEFI |
| "Disk too large for MBR" | >2TB | Gunakan GPT |

---

## ✅ Simpulan Kilat

| | | |
|---|---|---|
| Fitur | MBR | GPT |
| Maks Partisi | 4 primary (atau 3 + extended) | 128 primary |
| Disk Maksimum | 2 TB | > 9 ZB |
| Boot Mode | BIOS | UEFI |
| Backup Table | ❌ | ✅ |
| CRC Check | ❌ | ✅ |

---

Kalau lo mau gua buatin simulasi setup MBR vs GPT lengkap pake fdisk dan parted, tinggal bilang aja:

📣 “COBAIN MBR & GPT SEKALIGUS COACH!” 💪

🔥 Ngerti banget maksud lu bro — bukan cuma aturan umum MBR vs GPT, tapi aturan TEKNIS pas kita lagi bikin partisinya langsung, biar gak salah langkah saat ngoprek langsung di terminal!

---

# 📌 Aturan Saat Membuat Partisi (Real Praktek)

Berikut ini aturan dan best practice teknis pas lo lagi langsung bikin partisi pake fdisk, parted, atau GUI.

---

## 💣 [1] Jangan Tiban Partisi Aktif

- ❌ Jangan bikin partisi di atas partisi yang udah dipake (tanpa delete dulu).
- Bisa bikin korup data atau sistem gak bisa boot.

🧠 Solusi: wipefs, dd, atau hapus partisi dulu sebelum mkpart.

---

## 🧱 [2] Urutan Start-End Harus Benar

- 📏 Di fdisk atau parted, tentuin ukuran/posisi jangan sampai overlapping (bertabrakan).

🧠 Contoh Salah:

mkpart primary ext4 0% 50%

mkpart primary ext4 40% 100%  ❌ <– tabrakan 40%–50%

✅ Solusi:

mkpart primary ext4 0% 50%

mkpart primary ext4 50% 100%

---

## 🧩 [3] Jangan Lupa Format Setelah Buat Partisi

- Partisi itu baru "wadah".

- Harus diformat (ext4/xfs/fat32) dulu sebelum bisa dipakai.

✅ Contoh:

mkfs.ext4 /dev/sdb1

---

## 🔑 [4] Di MBR: Hanya 4 Partisi Primary!

- Maksimal 4 primary.
- Kalau butuh lebih, harus:
- bikin extended
- lalu baru logical di dalamnya

🧠 Contoh di fdisk:

n → primary → sda1

n → primary → sda2

n → primary → sda3

n → primary → sda4 (extended)

n → logical → sda5

---

## 🧠 [5] Di GPT: Bebas, Tapi Jangan Lupa Alignment!

- Parted otomatis align partisi ke blok disk (misal 1MiB)
- Tapi jangan pakai offset aneh kayak 1s (sector 1) → ini bisa tabrakan sama GPT header atau bootloader!

✅ Good example:

mkpart primary ext4 1MiB 512MiB

❌ Bad example:

mkpart primary ext4 1s 204800s  # bikin warning!

---

## 🔗 [6] Jangan Langsung Mount Disk Kosong

- Mount hanya berlaku untuk partisi, bukan disk mentah.

❌ Salah:

mount /dev/sdb /mnt

✅ Benar:

mount /dev/sdb1 /mnt

---

## 🚀 [7] Di UEFI: Butuh Partisi ESP (EFI System Partition)

- Ukuran: 100–512MB
- Format: FAT32
- Flag: boot, esp

🧠 Contoh parted (UEFI + GPT):

(parted) mkpart ESP fat32 1MiB 512MiB

(parted) set 1 boot on

(parted) set 1 esp on

---

## 📦 [8] Buat Partisi Swap? Ada Aturannya!

| | |
|---|---|
| RAM Size | Rekomendasi SWAP |
| <2GB | RAM × 2 |
| 2–8GB | RAM × 1 |
| >8GB | RAM × 0.5 |

🧠 Contoh Format Swap:

mkswap /dev/sdb2

swapon /dev/sdb2

## 🔥 [9] Jangan Lupa partprobe atau kpartx (reload partisi)

Setelah modifikasi partisi:

partprobe

Atau jika pake LVM:

kpartx -a /dev/sdb

---

## ✨ [10] Nama Partisi & Label Bantu Navigasi

Kasih nama label buat mempermudah mount/fstab:

e2label /dev/sdb1 DATA_DISK

---

## 🚀 Kesimpulan Praktis

| | | |
|---|---|---|
| Hal | MBR | GPT |
| Max partisi | 4 primary | 128 primary |
| Harus format? | ✅ | ✅ |
| Alignment sektor | Manual | Auto by parted |
| Extended/logical | Wajib kalau >4 | Gak perlu |
| Perlu ESP? | ❌ | ✅ (UEFI only) |

---

Kalau mau gua simulasiin dari awal kayak:

- Buat disk kosong
- Bikin 3 partisi dengan fdisk (MBR)
- Bikin 3 partisi GPT dengan parted
- Format + Mount + Tambah ke /etc/fstab

Cukup bilang:

🔥 “SIMULASI PRAKTEK FULL MBR & GPT COACH!” 🔥

Gua buatin langsung dari nol sampe bisa dipake 😎

🔥 Siapp bro, gua jelasin makna persen-persen (0%, 30%, 100%) waktu lo bikin partisi GPT pake parted, biar gak bingung pas praktek langsung!

---

## 🧠 Persen di parted Itu Apa?

parted support input ukuran dalam bentuk:

- Persentase (%)
- Ukuran absolut (MiB, GiB, s, sector)

Kalau pake %, artinya lo ngasih tahu posisi partisi relatif terhadap total kapasitas disk.

---

## 💡 Contoh Praktis: Disk 100GB

Misalnya lo punya disk /dev/sdb ukuran 100GB.

Trus lo jalanin:

(parted) mkpart primary ext4 0% 50%

✅ Artinya: Buat partisi dari 0% s.d. 50% kapasitas disk
➡️ Hasilnya: partisi 0–50GB

---

## 🔥 Real Example 3 Partisi

Misalnya kita bagi disk jadi 3 partisi sama besar:

(parted) mklabel gpt

(parted) mkpart primary ext4 0% 33%

(parted) mkpart primary ext4 33% 66%

(parted) mkpart primary ext4 66% 100%

✅ Artinya:

| | | | |
|---|---|---|---|
| Partisi | Start | End | Ukuran |
| sdb1 | 0% | 33% | 33% Disk |
| sdb2 | 33% | 66% | 33% Disk |
| sdb3 | 66% | 100% | 34% Disk |

---

## 🛠 Tips Praktek

1. Pake persen cocok buat ngebagi rata.
2. Kalau mau presisi, pake ukuran absolut:

(parted) mkpart primary ext4 1MiB 1024MiB

3. Jangan mulai dari 0s atau 1s, mulai dari 1MiB buat alignment yang bagus.

---

## ✅ Output Setelah lsblk

Misalnya hasil partisinya begini:

lsblk /dev/sdb

Output:

NAME  MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS

sdb  8:16  0 100G  0 disk

├─sdb1  8:17  0  33G  0 part

├─sdb2  8:18  0  33G  0 part

└─sdb3  8:19  0  34G  0 part

---

## 🤔 Kapan Gunakan % dan Kapan Gunakan Ukuran?

| | |
|---|---|
| Pakai % | Pakai Ukuran Absolut |
| Mau bagi rata | Mau presisi (misal 512MiB) |
| Cepat dan simpel | Partisi untuk boot/swap |
| Gak ribet mikir angka | UEFI, ESP, manual layout |

---

Kalau lu mau, bisa gua buatin simulasi GPT full pake persen dan satu lagi pake ukuran langsung, tinggal bilang aja:

“COBAIN DUA DUANYA COACH: % DAN MiB!”

🔥 Ready to go!

🔥 Oke coach! Sekarang kita bahas cara bikin partisi GPT dengan angka ukuran langsung (MiB, GiB, dsb), bukan lagi pakai persen %.

---

## 💡 Kenapa Pakai Angka?

Karena:

- Lu pengen ukuran pasti (misal: 512MiB buat EFI, 8GiB buat swap).
- Lebih presisi buat bootloader atau skema partisi yang lo rancang manual.
- Rekomendasi untuk UEFI + Linux biasanya:
- 512MiB → EFI System Partition (ESP)
- 1–2GiB → /boot
- 2–8GiB → swap
- sisanya → root (/) atau /home

---

## 🛠 Simulasi: Bikin Partisi GPT Pake Angka

Misal lo punya disk kosong di /dev/sdb, lakukan:

parted /dev/sdb

## Langkah-langkah:

(parted) mklabel gpt

(parted) unit MiB  # Set unit jadi MiB biar gampang

# EFI Partition

(parted) mkpart ESP fat32 1MiB 513MiB

(parted) set 1 esp on

# /boot partition

(parted) mkpart primary ext4 513MiB 2561MiB

# swap partition

(parted) mkpart primary linux-swap 2561MiB 6657MiB

# root /

(parted) mkpart primary ext4 6657MiB 100%

(parted) print

(parted) quit

---

## ✅ Hasil yang Didapat (lsblk):

Setelah selesai, lu bisa cek:

lsblk /dev/sdb

Output kira-kira:

NAME  MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS

sdb  8:16  0 100G  0 disk

├─sdb1  8:17  0  512M  0 part  # ESP

├─sdb2  8:18  0  2G  0 part  # /boot

├─sdb3  8:19  0  4G  0 part  # swap

└─sdb4  8:20  0 93.5G  0 part  # /

---

## ⚙️ Next Steps (Opsional):

Format partisi:

mkfs.fat -F32 /dev/sdb1

mkfs.ext4 /dev/sdb2

mkswap /dev/sdb3

mkfs.ext4 /dev/sdb4

1.

Mount:

mount /dev/sdb4 /mnt

mkdir /mnt/boot

mount /dev/sdb2 /mnt/boot

mkdir /mnt/boot/efi

mount /dev/sdb1 /mnt/boot/efi

2.

Aktifkan swap:

swapon /dev/sdb3

3.

---

## 🧠 Kesimpulan

| | |
|---|---|
| Unit | Artinya |

| | |
|---|---|
| MiB | Mega Binary (1024 * 1024 bytes) |
| GiB | Giga Binary (1024 MiB) |
| Lebih cocok dipakai | Saat install manual / EFI booting |

---

Kalau lo siap, kita bisa lanjut:

- ke LVM di atas partisi ini,
- install OS manual,
- atau mau encrypt-in partisi / pake LUKS juga bisa.

Tinggal bilang:

“Lanjut pasang LVM + OS bro!”
atau
“Encrypt disk GPT pake LUKS dong!”

🔥 Ready kapan aja, bro!

Yess bro, bener banget! MB ≠ MiB, GB ≠ GiB 😎
Biar gak bingung lagi pas ngatur partisi, install OS, atau ngitung disk space, gua jelasin sejelas-jelasnya nih 👇

---

## 📊 Perbedaan MB vs MiB, GB vs GiB

| Satuan | Nama Panjang | Ukuran Sesungguhnya | Basis | Dipakai Oleh |
|---|---|---|---|---|
| MB | Megabyte | 1 MB = 1,000,000 bytes | Desimal | Harddisk, vendor storage (Windows juga defaultnya) |
| MiB | Mebibyte | 1 MiB = 1,048,576 bytes ($1024^2$) | Biner | Sistem operasi (Linux/Unix), LVM, parted, fdisk |
| GB | Gigabyte | 1 GB = 1,000,000,000 bytes | Desimal | Umum di marketing storage |
| GiB | Gibibyte | 1 GiB = 1,073,741,824 bytes ($1024^3$) | Biner | OS Linux dan banyak software teknis |

## 🎯 Analogi Simple-nya:

Misal lo beli flashdisk 16 GB:

- Vendor bilang: 16 GB → artinya 16 x 1,000,000,000 = 16.000.000.000 bytes
- Tapi Linux atau Windows lo bisa baca:
  16.000.000.000 / 1.073.741.824 ≈ 14.9 GiB

Makanya keliatannya “kok ilang 1GB?” padahal bukan ilang, cuma beda satuan aja.

---

## 🧠 Kapan Gunakan MB/GB dan MiB/GiB?

| | |
|---|---|
| Mau Apa? | Gunakan |
| Ngatur partisi manual pakai parted | MiB / GiB (lebih presisi) |
| Baca ukuran harddisk/SSD di toko | MB / GB (standar marketing) |
| Ngitung RAM / swap | Biasanya pakai MiB / GiB |
| Pas bikin filesystem / LVM | MiB / GiB biar akurat |

---

## 🧪 Cek Sendiri di Linux

lsblk -b  # Baca ukuran dalam byte

lsblk -o NAME,SIZE,TYPE,MOUNTPOINTS

df -h  # default: GB/MB (desimal)

df -H  # kapital H: SI (desimal)

df -i  # inode info

---

## 🔥 Tips Cepat

- MB = $1000^2$ = 1.000.000 bytes

- MiB = 1024² = 1.048.576 bytes
- GB = 1000³ = 1.000.000.000 bytes
- GiB = 1024³ = 1.073.741.824 bytes

---

Kalau lo mau gua buatin script cek ukuran partisi dengan satuan desimal vs biner juga bisa bro, tinggal bilang aja:

“Bikinin script deteksi perbedaan MB dan MiB di partisi gue bro!”

Mau lanjut ke LVM atau partisi GPT+LUKS juga boleh!

Siap coachh! 😎
Sekarang gua tunjukin cara bikin partisi GPT pakai tool parted di Linux, dan gua bakal contohin dua gaya penulisan: pakai persen (%) dan angka (MiB) — terus kita lihat hasilnya kayak gimana di lsblk/parted.

---

## 💿 Contoh 1: Buat Partisi GPT pakai Persen (%)

parted /dev/sdb

(parted) mklabel gpt

(parted) mkpart primary ext4 0% 30%

(parted) mkpart primary ext4 30% 60%

(parted) mkpart primary ext4 60% 100%

(parted) quit

## 🔍 Penjelasan:

- 0% 30% = dari awal disk sampai 30% kapasitas.
- 30% 60% = partisi kedua dari 30% ke 60%.
- 60% 100% = sisa akhir disk.

## 📦 Hasil (cek pakai lsblk):

Misalnya /dev/sdb adalah disk 15GB, maka output:

lsblk

NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS

sdb      8:16   0   15G  0 disk

├─sdb1   8:17   0    4G  0 part

├─sdb2   8:18   0    4G  0 part

└─sdb3   8:19   0    7G  0 part

---

## 💿 Contoh 2: Buat Partisi GPT pakai Angka (MiB)

parted /dev/sdb

(parted) mklabel gpt

(parted) mkpart primary ext4 1MiB 512MiB

(parted) mkpart primary ext4 512MiB 4096MiB

(parted) mkpart primary ext4 4096MiB 100%

(parted) quit

## 🔍 Penjelasan:

- 1MiB 512MiB → partisi kecil buat boot misalnya.
- 512MiB 4096MiB → partisi tengah (sekitar 3.5GB)
- 4096MiB 100% → sisa space.

## 📦 Hasil lsblk:

NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS

sdb      8:16   0   15G  0 disk

├─sdb1   8:17   0  511M  0 part

├─sdb2   8:18   0  3.5G  0 part

└─sdb3   8:19   0   11G  0 part

---

## 🔥 Perbandingan Persen vs Angka

| | | |
|---|---|---|
| Gaya Input | Cocok Buat | Kelebihan |
| Persen (%) | Bagi disk secara merata | Simpel, cepat, ga perlu hitung |
| MiB | Kontrol presisi tinggi | Bisa disesuaikan ukuran teknis |

---

Kalau udah bikin partisinya, lanjut lo bisa:

- Format: mkfs.ext4 /dev/sdb1
- Mount: mount /dev/sdb1 /mnt/mydisk

Butuh gua buatin script otomatis partisi GPT full workflow juga bisa!

Mau lanjut ke LUKS, LVM, atau RAID sekalian partisinya bro?

Great question bro! 🔥 Sekarang gua bakal jelasin:

---

## 💿 Jenis-Jenis Partisi MBR (dan sedikit GPT)

### 🧱 1. Primary Partition (Partisi Primer)

📌 Pengertian:

- Partisi utama, bisa bootable.
- Maksimal cuma bisa 4 primary di MBR (tanpa extended).

🎯 Kapan Dipilih:

- Kalo lo cuma butuh 1-4 partisi aja.
- Biasanya digunakan buat:
- /boot
- / (root)
- Windows partition (NTFS)
- Sistem recovery partition

💉 Contoh:

(parted) mklabel msdos

(parted) mkpart primary ext4 1MiB 1025MiB

---

## 📦 2. Extended Partition (Partisi Perluasan)

📌 Pengertian:

- "Wadah" buat logical partition.
- Lo hanya bisa punya 1 extended partition.
- Extended gak bisa dipakai langsung buat data, isinya harus logical.

🎯 Kapan Dipilih:

- Lo pengen lebih dari 4 partisi di MBR.
- Lo punya banyak sistem/file system atau partisi swap tambahan.

🧪 Contoh:

(parted) mkpart extended 1025MiB 8192MiB

---

## 🧩 3. Logical Partition (Partisi Logikal)

📌 Pengertian:

- Partisi yang hidup di dalam extended partition.
- Bisa punya banyak logical partisi, lebih dari 4 total.

🎯 Kapan Dipilih:

- Lo udah make 3 primary + 1 extended.
- Sisanya masuk ke logical.
- Contoh buat:
- /home
- /var
- /opt
- swap

🧪 Contoh:

(parted) mkpart logical ext4 1026MiB 4096MiB

(parted) mkpart logical linux-swap 4096MiB 5120MiB

---

## ⚠️ Aturan Penting (MBR):

- Maks 4 Primary Partition.
- Bisa 3 Primary + 1 Extended → di dalam extended bisa banyak Logical.
- Logical harus di dalam Extended Partition.

---

## 📌 GPT Gimana?

- GPT gak butuh extended/logical, karena bisa langsung punya 128 partisi primary 😎
- Simpelnya: semua partisi di GPT adalah primary.
- Jadi, Extended/Logical cuma konsep di MBR.

---

## ✅ Tips Memilih:

| | |
|---|---|
| Kebutuhan | Pilihan |
| Cuma 1–4 partisi | Primary semua |
| Mau lebih dari 4 partisi (MBR) | Pakai Extended + Logical |
| Butuh kompatibilitas lama (BIOS, Windows) | MBR |
| Disk modern, UEFI, >2TB | GPT |

---

Mau gua buatin simulasi partisi lengkap:

- 2 primary
- 1 extended
- 2 logical
  di satu disk sdc misalnya? Atau lanjut bikin GPT + LVM sekalian? 😎

Oke broo! 😎🔥
Sekarang kita bakal bikin simulasi lengkap:
🧱 2 Primary, 📦 1 Extended, 🧩 2 Logical
semua di disk contoh: /dev/sdc
(Dianggap kosong total, belum ada partisi!)

---

# 🔧 Simulasi: Bikin Partisi MBR Manual di Linux

Tools: parted, tapi ntar dicek juga via lsblk atau fdisk -l

## 💣 1. Bikin Label Disk MBR (msdos)

parted /dev/sdc

(parted) mklabel msdos

---

## 🧱 2. Bikin 2 Primary Partition

### 📁 Primary 1 — Buat /boot (1G)

(parted) mkpart primary ext4 1MiB 1025MiB

### 📁 Primary 2 — Buat /root (4G)

(parted) mkpart primary ext4 1025MiB 5121MiB

---

## 📦 3. Bikin Extended Partition (4G–15G)

(parted) mkpart extended 5121MiB 15361MiB

---

## 🧩 4. Bikin 2 Logical Partition

### 📁 Logical 1 — Buat /home (6G)

(parted) mkpart logical ext4 5122MiB 11265MiB

### 📁 Logical 2 — Buat swap (3G)

(parted) mkpart logical linux-swap 11265MiB 14337MiB

---

## 🚪 5. Exit Parted

(parted) quit

---

## 🧪 Cek Hasilnya

lsblk /dev/sdc

Hasilnya bakal mirip:

sdc      8:32   0   15G  0 disk

├─sdc1   8:33   0    1G  0 part

├─sdc2   8:34   0    4G  0 part

├─sdc3   8:35   0   10G  0 part  (extended)

├─sdc5   8:36   0    6G  0 part  (logical)

└─sdc6   8:37   0    3G  0 part  (logical)

Note: sdc4 disk-nya gak keliatan karena slot sdc3 adalah extended, jadi logical mulai dari sdc5.

---

## 🛠️ 6. Format Filesystem

mkfs.ext4 /dev/sdc1

mkfs.ext4 /dev/sdc2

mkfs.ext4 /dev/sdc5

mkswap /dev/sdc6

## 🔗 7. Mount (contoh aja)

mkdir -p /mnt/boot /mnt/root /mnt/home

mount /dev/sdc1 /mnt/boot

mount /dev/sdc2 /mnt/root

mount /dev/sdc5 /mnt/home

swapon /dev/sdc6

---

## 🧠 Summary Table:

| | | | |
|---|---|---|---|
| Partisi | Tipe | Ukuran | Fungsi |
| sdc1 | Primary | 1G | /boot |
| sdc2 | Primary | 4G | / |
| sdc3 | Extended | 10G | Wadah |
| sdc5 | Logical | 6G | /home |
| sdc6 | Logical | 3G | swap |

---

Mau next simulasi:

- ✅ GPT layout?
- ✅ Buat partisi otomatis via fdisk?
- ✅ Integrasi ke /etc/fstab?
- ✅ Integrasi LVM langsung?

Lu yang pilih bro! 💪

Sip bro! Nih gua jelasin maksud, tujuan, dan cara kerja perintah yang lo tanyain itu secara deep dive tapi tetap santai 😎:

---

## 🧨 Perintah:

dd if=/dev/zero of=/dev/sdb bs=1M status=progress

## 🧠 Penjelasan per bagian:

| | |
|---|---|
| Bagian | Maksudnya |
| dd | Tool low-level untuk copy/overwrite data byte-per-byte |
| if=/dev/zero | Input File: isi dengan nol semua (0x00) dari device spesial /dev/zero |
| of=/dev/sdb | Output File: target-nya adalah disk /dev/sdb (!!! seluruh disk) |
| bs=1M | Block size: salin data per blok 1 Megabyte biar cepet |
| status=progress | Menampilkan progress saat berjalan (biar gak gelap-gelapan) |

---

## 🎯 Tujuan & Fungsi:

✅ Menghapus total isi disk — Ini menghancurkan semua isi di /dev/sdb:

- Header LUKS (encrypted metadata)
- Partisi (MBR/GPT)
- Isi data/logical volume
- Bahkan filesystem pun ikut lenyap

✅ Unenkripsi LUKS secara "brutal" — Karena header LUKS-nya kehapus, disk gak bisa dibuka lagi via cryptsetup open.

✅ Reset disk ke kondisi "kosong" seperti baru beli.

✅ Keamanan — Biasa dipakai buat:

- Wipe disk sebelum dijual
- Bersihin disk yang dipake buat VM, forensic, atau LUKS testing
- Ngilangin jejak rahasia 😌

---

## 💣 WARNING!

Hati-hati!
Lo literally ngebunuh isi disk total.
Setelah ini:

- Gak bisa undo
- Gak bisa di-recover
- Gak bisa buka partisi apapun

💀 NO GO BACK BRO

---

## 💡 Contoh Output:

2304059392 bytes (2.3 GB, 2.1 GiB) copied, 3.2354 s, 712 MB/s

---

## 🧪 Next Step:

Setelah lo wipe:

- Bisa langsung parted /dev/sdb → mklabel gpt atau mklabel msdos
- Lanjut buat partisi baru

---

Kalau lo pengen versi yang lebih aman/secure:

dd if=/dev/urandom of=/dev/sdb bs=1M status=progress

Itu pake data acak, bukan nol — jadi gak bisa ditebak sebelumnya disknya pernah isinya apa. Tapi jauh lebih lama.

---

Mau lanjut ke:

- 🔐 Secure wipe (DoD method)?
- 🔄 Restore MBR/GPT setelah di-wipe?
- 🧱 Build ulang disk dari nol?

Lo tentuin cuy 💪

🔥 Siap bro! Sekarang gua gabungin seluruh step setup LUKS + LVM lengkap sama hasil output lsblk & df -h-nya biar makin kebayang nyata kaya simulasi asli. Let's gooo! 😎💥

---

# 🎯 TUJUAN:

- Enkripsi disk baru /dev/sdb pakai LUKS
- Bangun LVM stack di atas disk terenkripsi (PV → VG → LV)
- Format & mount biar bisa dipakai
- Tampilkan hasil akhir lsblk dan df -h

---

# ✅ STEP-BY-STEP + CONTOH HASIL

---

## 🔹 1. Cek disk kosong dulu

lsblk

📄 Output awal:

NAME  MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS

sda     8:0    0   64G  0 disk

├─sda1  8:1    0    1G  0 part /boot

└─sda2  8:2    0   63G  0 part

sdb     8:16   0   15G  0 disk  <--- Disk baru kosong

---

## 🔹 2. Enkripsi disk /dev/sdb pakai LUKS

cryptsetup luksFormat /dev/sdb

Lalu isi passphrase.

---

## 🔹 3. Unlock disk terenkripsi

cryptsetup open /dev/sdb luks_simulasi

🔑 Akan muncul device baru:

/dev/mapper/luks_simulasi

---

## 🔹 4. Buat PV di dalam device terenkripsi

pvcreate /dev/mapper/luks_simulasi

---

## 🔹 5. Buat Volume Group

vgcreate vg_simulasi /dev/mapper/luks_simulasi

---

## 🔹 6. Buat Logical Volume

lvcreate -L 2G -n lv_data vg_simulasi

---

## 🔹 7. Format LV

mkfs.ext4 /dev/vg_simulasi/lv_data

---

## 🔹 8. Mount volume

mkdir /mnt/luks_simulasi

mount /dev/vg_simulasi/lv_data /mnt/luks_simulasi

---

## 🔹 9. Cek Hasil Akhir

### 📜 Output lsblk

lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS

```
sda                         8:0    0   64G  0 disk
├─sda1                      8:1    0    1G  0 part  /boot
└─sda2                      8:2    0   63G  0 part
  └─luks-root             253:0    0   63G  0 crypt
    ├─vg_main-root        253:1    0   40G  0 lvm   /
    └─vg_main-home        253:2    0   23G  0 lvm   /home
sdb                         8:16   0   15G  0 disk
└─luks_simulasi           253:4    0   15G  0 crypt
  └─vg_simulasi-lv_data   253:5    0    2G  0 lvm   /mnt/luks_simulasi
```

---

## 📜 Output df -h

df -h | grep luks_simulasi

/dev/mapper/vg_simulasi-lv_data  2.0G  6.0M  1.9G  1% /mnt/luks_simulasi

---

## ✅ HASIL AKHIR:

- Lo punya 1 disk terenkripsi penuh
- Di dalamnya ada LVM stack (VG + LV)
- Udah bisa lo gunain buat nyimpen file aman di /mnt/luks_simulasi

---

## 🎁 BONUS NEXT STEP:

Kalau lo mau lanjut:

- 🔁 Auto-mount saat boot (via /etc/crypttab & /etc/fstab)
- 🔄 Resize encrypted volume
- 🔐 Tambah luksKeyFile buat dual unlock method

Tinggal bilang aja, lanjut kita bahas di next stage 🔥

Mantap cuy! 🔥 Kita bahas cara kerja LVM plus contoh praktik langsung biar makin nyantol di otak, dari PV → VG → LV, lengkap sama penjelasan maksud, tujuan, dan hasil akhirnya. Gaskeun! 🚀

---

# 🧠 Konsep Umum LVM:

| | | |
|---|---|---|
| Komponen | Fungsi | Tujuan |
| PV (Physical Volume) | Menjadikan disk/partisi siap dipakai LVM | Mengelola storage dari fisik ke level virtual |
| VG (Volume Group) | Gabungan dari PV jadi 1 pool storage | Bisa gabungin banyak disk jadi satu kesatuan |
| LV (Logical Volume) | Volume yang bisa di-mount ke sistem | Bisa dibuat, diubah ukuran, dipindah fleksibel |

---

# 🔧 PRAKTIK LANGSUNG

Misal kita pakai /dev/sdb (disk kosong).

## 🔹 1. Physical Volume (PV)

📌 Tujuan: Tandain disk biar bisa dipakai LVM.

pvcreate /dev/sdb

📜 Output:

Physical volume "/dev/sdb" successfully created.

Cek hasil:

pvs

📜 Output:

PV VG Fmt Attr PSize PFree

/dev/sdb lvm2 --- 15.00g 15.00g

## 🔹 2. Volume Group (VG)

📌 Tujuan: Gabungin PV jadi 1 tempat besar untuk bikin LV.

vgcreate vg_data /dev/sdb

📜 Output:

Volume group "vg_data" successfully created

Cek hasil:

vgs

📜 Output:

VG #PV #LV #SN Attr VSize VFree

vg_data 1 0 0 wz--n- 15.00g 15.00g

---

## 🔹 3. Logical Volume (LV)

📌 Tujuan: Bikin volume fleksibel dari VG.

lvcreate -L 5G -n lv_backup vg_data

📜 Output:

Logical volume "lv_backup" created.

Cek hasil:

lvs

📜 Output:

LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert

lv_backup vg_data -wi-a----- 5.00g

---

## 🔹 4. Format & Mount LV

📌 Tujuan: Siapin volume agar bisa dipakai oleh OS.

mkfs.ext4 /dev/vg_data/lv_backup

mkdir /mnt/backup

mount /dev/vg_data/lv_backup /mnt/backup

Cek:

df -h | grep backup

🧻 Output:

/dev/mapper/vg_data-lv_backup  5.0G   24M  4.9G   1% /mnt/backup

---

# 🎯 HASIL AKHIR

lsblk

🧻 Output:

sdb                     8:16   0   15G  0 disk

└─vg_data-lv_backup     253:0    0    5G  0 lvm   /mnt/backup

📦 VG = vg_data berisi LV = lv_backup sebesar 5GB
📁 Sudah dimount ke /mnt/backup dan siap dipakai

---

# ✨ Kelebihan LVM:

✅ Bisa resize volume:

- Tambah: lvextend
- Kurangi: lvreduce

✅ Bisa pindah antar disk (pakai pvmove)

✅ Cocok buat server/VM yang sering upgrade kapasitas storage

---

Kalau lo mau lanjut simulasi resize LV, pindah PV, atau gabungin 2 disk ke 1 VG, tinggal bilang aja cuy — langsung kita gas 🔥

Pertanyaan lu 🔥 banget bro! Jadi gini, kita bahas di mana posisi LVM (PV, VG, LV) dalam jenis-jenis partisi seperti:

- 🧱 Primary
- 📦 Extended
- 🧩 Logical

dan hubungannya dengan struktur LVM.

---

## 💡 Jawaban Singkat:

LVM itu berdiri di atas partisi yang kamu buat dulu di disk (entah itu partisi primer atau logikal). Setelah partisi jadi, baru LVM digunakan di dalamnya (mulai dari pvcreate, vgcreate, lvcreate).

---

## 🔍 Penempatan LVM pada Jenis Partisi

| Jenis Partisi | Bisa Jadi PV? | Cocok Buat LVM? | Catatan |
|---|---|---|---|
| Primary Partition | ✅ YA | ✅ YA | Biasanya dipakai kalau cuma butuh 1-2 LVM |
| Extended Partition | ❌ GAK BISA | ❌ | Hanya kontainer buat partisi logikal |
| Logical Partition | ✅ YA | ✅ YA | Bisa dipakai LVM sama seperti primary |

---

## 💡 Contoh:

Misalnya lo punya disk /dev/sdc

## 📘 1. LVM pakai Primary Partition:

parted /dev/sdc mklabel msdos

parted /dev/sdc mkpart primary 1MiB 8GiB

pvcreate /dev/sdc1

vgcreate vg_primary /dev/sdc1

lvcreate -L 2G -n lv_utama vg_primary

📌 /dev/sdc1 = partisi primer
📌 LVM dibikin di dalam partisi tersebut

---

## 📘 2. LVM pakai Logical Partition:

parted /dev/sdc mklabel msdos

parted /dev/sdc mkpart primary 1MiB 2GiB  # Partisi primer

parted /dev/sdc mkpart extended 2GiB 100%  # Partisi extended

parted /dev/sdc mkpart logical 2GiB 10GiB  # Partisi logikal

pvcreate /dev/sdc5

vgcreate vg_logical /dev/sdc5

lvcreate -L 1G -n lv_logik vg_logical

📌 /dev/sdc5 = partisi logikal (note: biasanya dia mulai dari 5)
📌 Tetap bisa jadi PV → VG → LV

---

## 📘 3. Kalau pakai GPT?

✅ GPT tidak ada batasan primary vs extended vs logical.
Semua partisinya langsung bisa jadi PV. Contoh:

parted /dev/sdc mklabel gpt

parted /dev/sdc mkpart primary 1MiB 100%

pvcreate /dev/sdc1

vgcreate vg_gpt /dev/sdc1

lvcreate -L 3G -n lv_gpt vg_gpt

---

## 🎯 Kesimpulan Penting

| | | | |
|---|---|---|---|
| Tipe Disk | Struktur Partisi | Bisa Pakai LVM? | Note |
| MBR | Primary | ✅ | Max 4 partisi utama |
| MBR | Logical (dalam extended) | ✅ | Buat lebih dari 4 partisi |
| GPT | Semua langsung primary | ✅ | Recommended buat sistem modern |

---

Kalau mau lanjut ke simulasi LVM di atas partisi primer dan logikal secara bersamaan, gua siap bantu full script + hasil lsblk juga 🔧💪

Wih mantap broo, kita analogikan konsep LVM (PV, VG, LV) biar makin nempel di otak kayak lem alteco 🔥

---

## 🧠 Analogi Simpel LVM

Bayangin lu lagi mau bangun dapur warteg 💡

| | | |
|---|---|---|
| Komponen LVM | Analogi Warteg | Penjelasan |
| PV (Physical Volume) | Galon-galon berisi beras, minyak, air | Disk fisik (partisi) yang lu siapin buat disatuin |
| VG (Volume Group) | Gudang warteg tempat semua bahan disimpen | Gabungan galon-galon (PV) jadi satu tempat penyimpanan besar |
| LV (Logical Volume) | Panci-panci buat masak nasi, sayur, dll | Isi dari gudang tadi yang dibagi sesuai kebutuhan (volume yang bisa di-mount dan dipakai OS) |

## 🧱 Contoh Praktiknya di Linux

Misalnya kita pakai disk /dev/sdc, dan kita pengen simulasikan analogi tadi 👇

---

## ✅ 1. Bikin Galon = PV

pvcreate /dev/sdc

📦 Artinya: "Siapkan galon buat dipakai masukin ke gudang (VG)"

---

## ✅ 2. Bikin Gudang = VG

vgcreate gudang_warteg /dev/sdc

📦 Artinya: "Masukin galon ke gudang. Sekarang kita punya tempat nyimpen beras dll"

---

## ✅ 3. Bikin Panci = LV

lvcreate -L 2G -n panci_sayur gudang_warteg

📦 Artinya: "Ambil 2 liter dari gudang buat masak sayur pake panci"

---

## ✅ 4. Format & Mount

mkfs.ext4 /dev/gudang_warteg/panci_sayur

mkdir /mnt/warteg

mount /dev/gudang_warteg/panci_sayur /mnt/warteg

📦 Artinya: "Panci sayur siap digunakan, udah ditaruh di kompor (mount point)"

---

## 📊 Hasil Akhir: lsblk

lsblk

sdc 8:32 0 10G 0 disk

└─gudang_warteg-panci_sayur 253:0 0 2G 0 lvm /mnt/warteg

---

# 🎯 Tujuan LVM secara umum:

| | |
|---|---|
| Fitur | Penjelasan |
| Fleksibel | Bisa resize LV kapan aja tanpa ngaruh ke disk fisik |
| Expandable | Tambah disk baru? Gabungin aja ke VG, beres |
| Snapshot | Bisa backup LV dengan cepat |
| Migrasi Mudah | Bisa pindah volume antar disk tanpa matiin sistem |

---

Kalau mau gua buatin simulasi LVM seperti ini dari nol langsung + hasilnya, tinggal bilang:

💡 "Gas buatin simulasinya bro, dari awal disk kosong sampe mount!"

💪

🔥 Mantap bro! Jadi hasil akhir lsblk yang nunjukin struktur LVM seperti ini:

sdc 8:32 0 10G 0 disk

└─gudang_warteg-panci_sayur 253:0 0 2G 0 lvm /mnt/warteg

Itu bukan langsung muncul gitu aja ya, tapi hasil dari rangkaian perintah sebelumnya yang lu jalankan bertahap. Nih gua breakdown perintah mana aja yang bikin itu bisa muncul:

---

## ✅ 1. Bikin Physical Volume (PV)

pvcreate /dev/sdc

Ini ngebuat disk /dev/sdc jadi bisa dipakai buat LVM, alias jadi "galon" penyimpanan pertama.

---

## ✅ 2. Bikin Volume Group (VG)

vgcreate gudang_warteg /dev/sdc

Ini bikin gudang penyimpanan dari PV tadi. Nama gudangnya: gudang_warteg.

---

## ✅ 3. Bikin Logical Volume (LV)

lvcreate -L 2G -n panci_sayur gudang_warteg

Ini bikin volume 2GB dari gudang tadi. Jadi LV baru dengan nama: panci_sayur.

---

## ✅ 4. Format LV biar bisa dipakai

mkfs.ext4 /dev/gudang_warteg/panci_sayur

Format si panci pake filesystem ext4, biar bisa dipake sistem Linux kayak biasa.

---

## ✅ 5. Mount ke sistem

mkdir /mnt/warteg

mount /dev/gudang_warteg/panci_sayur /mnt/warteg

Ini naruh panci ke atas kompor. Mount si LV ke folder supaya bisa diakses.

---

## ✅ 6. Liat hasilnya (di lsblk)

lsblk

Nah ini dia yang akhirnya nampilin semua struktur disk, VG, LV, dan mount point-nya.

---

## 🧠 Rangkuman Singkat:

| Tahap | Perintah |
|---|---|
| Buat PV | pvcreate /dev/sdc |
| Buat VG | vgcreate gudang_warteg /dev/sdc |
| Buat LV | lvcreate -L 2G -n panci_sayur gudang_warteg |
| Format | mkfs.ext4 /dev/gudang_warteg/panci_sayur |
| Mount | mkdir /mnt/warteg && mount /dev/gudang_warteg/panci_sayur /mnt/warteg |
| Lihat Struktur | lsblk |

---

Kalau mau lanjut simulasinya sampe tahap resize, snapshot, atau mount otomatis via fstab, gas aja bro tinggal bilang:
"Lanjutkan simulasinya ke fitur-fitur LVM yang keren bro!" 🧨

Great question bro! 🔥 Ini nih topik penting yang sering bikin bingung — bedanya LVM vs partisi biasa (MBR/GPT), dan kapan lo harus pakai yang mana. Nih gua kupas tuntas secara santai tapi padat:

---

## ⚔️ BEDANYA: LVM vs Partisi Biasa

| Aspek | Partisi Biasa (MBR/GPT) | LVM (Logical Volume Manager) |
|---|---|---|
| 📦 Struktur | Statis, langsung dari disk fisik | Dinamis, virtual di atas PV/VG |
| 🔄 Resize | Susah, kadang harus unmount & reboot | Bisa resize online (lvextend/lvreduce) |
| 🔗 Gabung Disk | Gak bisa gabungin beberapa disk | Bisa gabung banyak disk jadi satu VG |
| 📁 Flexibilitas | Fix ukuran dari awal | Bisa diubah ukuran kapan aja |
| 💣 Risiko Kesalahan | Kalau salah resize bisa corrupt partisi | Lebih aman dan fleksibel |
| 🧰 Tools | fdisk, parted | pvcreate, vgcreate, lvcreate |
| 📍 Mount Point | Mount langsung ke partisi (/dev/sda1) | Mount ke volume logis (/dev/vg/lv) |

## 🎯 KAPAN PAKAI PARTISI BIASA?

✅ Cocok untuk:

- Sistem kecil, sederhana (misalnya Raspberry Pi, laptop basic).
- Dual-boot manual (Windows & Linux).
- Buat partisi /boot, EFI, swap (yang butuh fix location).

🧠 Contoh:

parted /dev/sda

mklabel gpt

mkpart primary ext4 0% 50%

mkpart primary ext4 50% 100%

---

## 🚀 KAPAN PAKAI LVM?

✅ Cocok untuk:

- Server, VM, storage besar (misal: Proxmox, RHEL, Ubuntu Server).
- Butuh resize disk tanpa reboot.
- Backup pakai snapshot.
- Gabung beberapa disk jadi satu.

🧠 Contoh:

pvcreate /dev/sdb

vgcreate data_vg /dev/sdb

lvcreate -L 10G -n backup_lv data_vg

mkfs.ext4 /dev/data_vg/backup_lv

mount /dev/data_vg/backup_lv /mnt/backup

---

# 🧠 ANALOGI GAMPANGNYA:

Partisi Biasa = Kotak makan plastik
Lo beli kotak 3 sekat. Mau ganti ukuran? Harus beli baru.

LVM = Bento Box Modular
Kotaknya bisa lo tambah-sekat, lo geser, lo gabung sama bento lain, semua fleksibel.

---

# 📌 Kombinasi Ideal (Best Practice)

Biasanya di sistem production:

/boot → partisi biasa (non-LVM)

/ → LVM (LV1)

/home → LVM (LV2)

/var/log → LVM (LV3)

---

Kalau lo bikin simulasi LVM, lo bakal nemu benefit-nya pas:

- Resize (pakai lvextend, resize2fs)
- Backup pakai snapshot (lvcreate --snapshot)
- Gabung disk kecil jadi 1 VG

---

Kalau lo mau gua contohin semua perbandingan langsung (bikin partisi biasa vs LVM, terus lihat di lsblk dan manfaatnya langsung), tinggal bilang:

"Simulasikan head-to-head LVM vs partisi biasa bro!"
Gas lanjut 😎

Nice bro! 😎 Langsung gua simulasikan perbandingan LVM vs Partisi Biasa (GPT) dengan output lsblk akhir, biar lo ngerti head-to-head comparison mereka. Gua pakai disk baru misalnya /dev/sdc.

---

# 🧱 1. SIMULASI: PARTISI BIASA (GPT)

---

## 📝 LANGKAH-LANGKAH:

# Masuk ke parted

parted /dev/sdc

# Buat GPT label

(parted) mklabel gpt

# Buat partisi biasa (pakai persen)

(parted) mkpart primary ext4 0% 50%

(parted) mkpart primary ext4 50% 100%

(parted) quit

# Format

mkfs.ext4 /dev/sdc1

mkfs.ext4 /dev/sdc2

# Mount

mkdir -p /mnt/gpt1 /mnt/gpt2

mount /dev/sdc1 /mnt/gpt1

mount /dev/sdc2 /mnt/gpt2

---

## 📊 HASIL lsblk:

lsblk

sdc 8:32 0 10G 0 disk

├─sdc1  8:33  0  5G  0 part  /mnt/gpt1

└─sdc2  8:34  0  5G  0 part  /mnt/gpt2

---

## 💡 KESIMPULAN (PARTISI BIASA):

- Lo bikin langsung dari disk (langsung /dev/sdc1, /dev/sdc2).
- Nggak bisa resize partisi seenaknya.
- Gak bisa gabungin disk lain ke dalam partisi.
- Simpel, tapi terbatas.

---

## 🔐 2. SIMULASI: LVM + LUKS (ADVANCED SETUP)

---

### 📝 LANGKAH-LANGKAH:

# Hancurin dulu isi disk sdc

dd if=/dev/zero of=/dev/sdc bs=1M status=progress

# Enkripsi

cryptsetup luksFormat /dev/sdc

cryptsetup open /dev/sdc luks_simulasi

# Buat LVM

pvcreate /dev/mapper/luks_simulasi

vgcreate vg_warteg /dev/mapper/luks_simulasi

lvcreate -L 3G -n lv_sayur vg_warteg

lvcreate -L 2G -n lv_nasi vg_warteg

# Format & mount

mkfs.ext4 /dev/vg_warteg/lv_sayur

mkfs.ext4 /dev/vg_warteg/lv_nasi

mkdir -p /mnt/sayur /mnt/nasi

mount /dev/vg_warteg/lv_sayur /mnt/sayur

mount /dev/vg_warteg/lv_nasi /mnt/nasi

---

## 📊 HASIL lsblk:

lsblk

sdc 8:32 0 10G 0 disk

└─luks_simulasi 253:0 0 10G 0 crypt

├─vg_warteg-lv_sayur 253:1 0 3G 0 lvm /mnt/sayur

└─vg_warteg-lv_nasi 253:2 0 2G 0 lvm /mnt/nasi

---

## 💡 KESIMPULAN (LVM + LUKS):

- Disk dienkripsi, aman dari pencurian data.
- Bisa resize logical volume kapan aja.
- Fleksibel, bisa gabung disk lain ke VG.
- Mount-nya ke LV (logical volume), bukan partisi biasa.

---

## ⚖️ HEAD-TO-HEAD:

| | | |
|---|---|---|
| Fitur | Partisi Biasa | LVM + LUKS |
| 🔐 Enkripsi | Tidak | Ya (LUKS) |
| 📦 Resize | Susah | Mudah (lvextend/lvreduce) |
| 📁 Mount | /dev/sdc1, /dev/sdc2 | /dev/vg/lv_name |

| | | |
|---|---|---|
| 🧠 Fleksibilitas | Terbatas | Tinggi |
| 📊 Output lsblk | sdc1, sdc2 | vg-lv via /dev/mapper |

---

Kalau lo kerja di dunia Cyber Security, Server Admin, Cloud, 99% pasti pake LVM + LUKS. Tapi buat sistem sederhana, partisi biasa udah cukup.

Kalau mau gua kasih versi perintah otomatis full script bash-nya, tinggal bilang aja:

"Bungkus jadi script dong bro, tinggal tempel langsung run" 😎

Yes, bener banget bro! 💯

Kalau lo pakai LUKS, maka data yang ada di dalam partisi lo bakal dijaga ketat lewat enkripsi full disk. Nih gua breakdown biar makin paham:

---

# 🔐 Fungsi LUKS (Linux Unified Key Setup)

## ➤ Apa yang dijaga LUKS?

LUKS nge-enkripsi seluruh isi disk atau partisi yang lo pilih. Artinya:

- Isi filesystem, file, folder, bahkan struktur disk lo, semuanya jadi gak bisa dibaca tanpa passphrase.
- Begitu power dimatiin, semua data yang tadi "terbuka" bakal kembali terenkripsi.

---

# 🔒 Contoh Perlindungan LUKS

## Misalnya lo punya data penting di /dev/sdb:

cryptsetup luksFormat /dev/sdb  # Enkripsi disk

cryptsetup open /dev/sdb securedata  # Unlock dan muncul sebagai /dev/mapper/securedata

mkfs.ext4 /dev/mapper/securedata  # Format (isi baru)

mount /dev/mapper/securedata /mnt  # Pakai disk-nya

📦 Data lo disimpan di /mnt. Tapi...

## 🚨 Kalau ada yang nyolong harddisk lo:

- Mereka cuma lihat isi /dev/sdb = acak-acakkan (ciphertext)
- Gak bisa dibuka tanpa passphrase
- Walau dibongkar pakai tools forensic, tetap gak bisa dibaca tanpa kunci yang sah

---

## ✅ Keuntungan Pakai LUKS

| | |
|---|---|
| Fitur | Ada di LUKS |
| Enkripsi AES-256 Kuat | ✅ |
| Bisa multi-passphrase (slot) | ✅ |
| Perlindungan saat dicuri | ✅ |
| Compatible sama LVM | ✅ |

---

## 🚫 Kelemahan (yang harus diperhatikan)

- Kalau passphrase hilang, data gak bisa dibalikin.
- Enkripsi butuh sedikit lebih banyak resource CPU (tapi modern CPU udah punya AES-NI hardware accel).

---

## 🔥 Analogi Simpel:

LUKS itu kayak brankas super canggih. Lo bisa taro file apapun di dalamnya, tapi pas keluar dari rumah, brankas langsung auto-lock, dan cuma lo yang bisa buka pake kode yang lo buat.

---

Kalau lo pengen gua simulasikan kasus serangan dan buktiin gimana LUKS nahan akses, gua bisa bantu juga. Mau? 😎

🔥 Bisa bro! Tapi ada perbedaan alur dan fleksibilitasnya. Jadi lo bisa pakai LUKS di partisi biasa juga, gak harus pakai LVM, cuma ada perbedaan power & scalability-nya.

---

## ✅ LUKS di Partisi Biasa — YES, ITU BISA

Contoh skenario:

Lo cuma butuh satu partisi terenkripsi tanpa ribet LVM-LVM-an.

---

### 📃 Contoh: Enkripsi langsung partisi biasa /dev/sdc1

# 1. Bikin disk label dan partisi biasa

parted /dev/sdc mklabel gpt

parted /dev/sdc mkpart primary ext4 0% 100%

# 2. Enkripsi si partisi biasa (bukan disk penuh)

cryptsetup luksFormat /dev/sdc1

# 3. Unlock

cryptsetup open /dev/sdc1 luks_plainpart

# 4. Format dan mount

mkfs.ext4 /dev/mapper/luks_plainpart

mkdir /mnt/plainpart

mount /dev/mapper/luks_plainpart /mnt/plainpart

---

### 📊 Hasil lsblk:

lsblk

sdc 8:32 0 10G 0 disk

```
└─sdc1        8:33   0   10G  0 part
  └─luks_plainpart
                253:0    0   10G  0 crypt /mnt/plainpart
```

---

## 🔄 BEDANYA APA DENGAN LVM + LUKS?

| Fitur | LUKS + Partisi Biasa | LUKS + LVM |
|---|---|---|
| 🔐 Enkripsi | ✅ Ya | ✅ Ya |
| 📦 Resize | ❌ Susah (resize2fs + parted) | ✅ Mudah (lvextend) |
| 📀 Multi-volume di dalam disk | ❌ Gak bisa | ✅ Bisa (LV banyak dalam 1 VG) |
| 📁 Mount | Langsung satu device | Fleksibel (per LV mountable) |
| 🧠 Manajemen storage | Basic | Advanced |

---

## 🎯 Kapan LUKS + Partisi Biasa COCOK?

- Kalo lo cuma mau enkripsi satu partisi aja, buat:
- Backup drive terenkripsi
- USB flash drive aman
- SSD portable buat kerja

## 🎯 Kapan LUKS + LVM COCOK?

- Kalo lo setup:
- Server
- Laptop kerja multi-volume
- Simulasi untuk dunia kerja IT, Sysadmin, atau LKS 😎

---

## 🧠 Analogi Gampang:

| Konsep | Analogi |
| --- | --- |
| LUKS + Partisi | Satu brankas besar langsung jadi lemari |
| LUKS + LVM | Brankas → lemari fleksibel, isi rak-rak banyak yang bisa dibagi-bagi ukuran dan ditambah |

---

Kalau mau gua bikinin 2 simulasi berdampingan langsung:

1. LUKS di partisi biasa
2. LUKS di LVM

Dengan hasil lsblk + struktur file system, tinggal bilang:

"Bungkus simulasi keduanya sekalian bro, biar gua compare langsung" 🧠🔥

Sip bro! Nih gua bungkus lengkap dua simulasi head-to-head:

---

# 🧪 SIMULASI:

## 🔐 A. LUKS + Partisi Biasa

## 🧠 B. LUKS + LVM (yang advance dan fleksibel)

---

## 🔐 A. LUKS + PARTISI BIASA (Tanpa LVM)

### 💡 Tujuan:

- Enkripsi satu partisi biasa pakai LUKS
- Tanpa manajemen fleksibel (resize, multiple volume, dll)

---

### 📝 LANGKAH-LANGKAH:

# 1. Bikin GPT dan partisi

parted /dev/sdc mklabel gpt

parted /dev/sdc mkpart primary ext4 0% 100%

# 2. Enkripsi partisinya

cryptsetup luksFormat /dev/sdc1

# 3. Buka (unlock) partisinya

cryptsetup open /dev/sdc1 luks_plain

# 4. Format partisi terenkripsi

mkfs.ext4 /dev/mapper/luks_plain

# 5. Mount ke sistem

mkdir /mnt/plain

mount /dev/mapper/luks_plain /mnt/plain

---

## 📊 HASIL lsblk:

lsblk

sdc 8:32 0 10G 0 disk

└─sdc1 8:33 0 10G 0 part

└─luks_plain 253:0 0 10G 0 crypt /mnt/plain

---

## ✅ KESIMPULAN:

- Enkripsi 1 partisi = aman ✅
- Gak fleksibel: 1 partisi = 1 filesystem
- Gak bisa resize LV, gak bisa multi-volume

## 🧠 B. LUKS + LVM (Power Mode)

### 💡 Tujuan:

- Enkripsi disk penuh pakai LUKS
- Bikin banyak Logical Volume (LV) dalam 1 enkripsi
- Lebih fleksibel, resize/gabung gampang

---

### 📝 LANGKAH-LANGKAH:

# 1. Hapus isi disk (opsional)

dd if=/dev/zero of=/dev/sdc bs=1M status=progress

# 2. Enkripsi disk langsung

cryptsetup luksFormat /dev/sdc

# 3. Unlock disk

cryptsetup open /dev/sdc luks_full

# 4. Siapkan LVM

pvcreate /dev/mapper/luks_full

vgcreate vg_advanced /dev/mapper/luks_full

lvcreate -L 6G -n lv_data vg_advanced

lvcreate -L 3G -n lv_backup vg_advanced

# 5. Format volume

mkfs.ext4 /dev/vg_advanced/lv_data

mkfs.ext4 /dev/vg_advanced/lv_backup

# 6. Mount

```
mkdir -p /mnt/data /mnt/backup

mount /dev/vg_advanced/lv_data /mnt/data

mount /dev/vg_advanced/lv_backup /mnt/backup
```

---

## 📊 HASIL lsblk:

```
lsblk

sdc                      8:32   0   10G  0 disk
└─luks_full              253:0   0   10G  0 crypt
  ├─vg_advanced-lv_data    253:1   0    6G  0 lvm   /mnt/data
  └─vg_advanced-lv_backup  253:2   0    3G  0 lvm   /mnt/backup
```

---

## ✅ KESIMPULAN:

- Lebih fleksibel & scalable
- Bisa bikin banyak volume
- Bisa resize dan migrasi volume

---

## ⚔️ HEAD-TO-HEAD COMPARISON

| | | |
|---|---|---|
| Fitur | LUKS + Partisi Biasa | LUKS + LVM |
| Enkripsi | ✅ Ya | ✅ Ya |
| Jumlah Volume | 1 partisi aja | Banyak LV fleksibel |
| Resize Volume | Susah | ✅ Mudah pakai lvextend/lvreduce |
| Management Disk | Manual | Otomatis via VG dan LV |
| Output lsblk | /dev/mapper/luks_plain | /dev/vg_advanced/lv_name |
| Cocok Buat | Drive sederhana, backup | Server, laptop kerja, cloud |

---

Kalau lo mau:

- 🔐 Simple, aman, satu partisi doang → LUKS + partisi biasa
- 💡 Fleksibel, powerful, scalable → LUKS + LVM

---

Kalo lo butuh:

💥 "Script otomatis tinggal run semua setup ini"
Gua siap buatin bro tinggal bilang aja:
"Bungkus script bash-nya bro!" 🧠🔥

nmcli con add type ethernet ifname (example : enp0s8) con-name hostonly dhcp4 yes

nmcli con up hostonly

**