BtrFS Installation Kubuntu 25.10

Dann machen wir es so automatisch wie unter Kubuntu eben möglich.
Du klickst im Prinzip *nur noch den Installer durch*, der Rest sind zwei Skripte.

Ich schlage vor:

1. **Prepare-Script** im Live-System → macht GPT-Layout + Btrfs-Partition
2. Kubuntu ganz normal installieren (1–2 Klicks bei der Partitionierung)
3. **Post-Install-Script** im Live-System → richtet Subvolumes + `fstab` vollautomatisch ein

Beide Skripte sind **`sgdisk`-frei** (nur `parted` + `btrfs`).

1️⃣ Prepare-Script (vor der Installation)

Zweck: Platte leeren, GPT anlegen, EFI + Btrfs-Partition erstellen.
Noch keine Subvolumes, die kommen nach der Installation.

Passe oben **`DISK`** an (z. B. `/dev/nvme0n1` oder `/dev/sda`).

```
#!/usr/bin/env bash
set -euo pipefail

### === KONFIGURATION ANPASSEN ===
DISK="/dev/sdX"             # z.B. /dev/nvme0n1 oder /dev/sda
EFI_SIZE_MIB=512            # Größe der EFI-Partition in MiB
BTRFS_LABEL="KUBUNTU_ROOT"  # Label für das Btrfs-Dateisystem

echo ">>> WARNUNG: $DISK wird KOMPLETT GELÖSCHT!"
read -p "Weiter mit ENTER, Abbrechen mit STRG+C..."

### 1. Alle alten Signaturen entfernen
sudo wipefs -a "$DISK"
```

```
### 2. GPT + Partitionen anlegen (nur parted, kein sgdisk)
sudo parted -s "$DISK" \
  mklabel gpt \
  unit MiB \
  mkpart EFI fat32 1 $((1 + EFI_SIZE_MIB)) \
  set 1 esp on \
  mkpart ROOT btrfs $((1 + EFI_SIZE_MIB)) 100%

EFI_PART="${DISK}1"
ROOT_PART="${DISK}2"

echo ">>> Neue Partitionen:"
lsblk "$DISK"

### 3. Dateisysteme anlegen
echo ">>> Formatiere EFI-Partition ($EFI_PART) als FAT32..."
sudo mkfs.vfat -F32 -n EFI "$EFI_PART"

echo ">>> Formatiere ROOT-Partition ($ROOT_PART) als Btrfs..."
sudo mkfs.btrfs -f -L "$BTRFS_LABEL" "$ROOT_PART"

echo
echo "=== SCHRITT IM INSTALLER (KUBUNTU 25.10) ==="
echo "1) 'Etwas Anderes' / manuelle Partitionierung wählen."
echo "2) $EFI_PART:"
echo "     - Typ: EFI System Partition"
echo "     - Dateisystem: FAT32"
echo "     - Einhängepunkt: /boot/efi"
echo "     - NICHT neu formatieren."
echo "3) $ROOT_PART:"
echo "     - Dateisystem: Btrfs"
echo "     - Einhängepunkt: /"
echo "     - NICHT neu formatieren."
echo
echo "Dann Installation ganz normal durchlaufen lassen."
```

Benutzung:

1. Kubuntu-Live-ISO booten
2. Terminal öffnen, Script z. B. als `prepare-btrfs.sh` speichern, ausführbar machen:

```
chmod +x prepare-btrfs.sh
./prepare-btrfs.sh
```

3. Danach Kubuntu-Installer starten und die Hinweise aus der Ausgabe befolgen.

---

2 Post-Install-Script (nach der Installation)

Zweck:

- vorhandene Kubuntu-Installation auf Btrfs-Subvolumes umziehen
- `@`, `@home`, `@log`, `@cache`, `@snapshots` anlegen
- `/etc/fstab` automatisch anpassen

Das machst du **am bequemsten wieder vom Live-System aus**, nachdem Kubuntu fertig installiert wurde.

Oben **`ROOT_PART`** auf deine Btrfs-Partition setzen (z. B. `/dev/nvme0n1p2`).

```
#!/usr/bin/env bash
set -euo pipefail

### === ANPASSEN: Btrfs-Root-Partition der Kubuntu-Installation ===
ROOT_PART="/dev/nvme0n1p2"

TOP_MNT="/mnt/btrfs-top"
NEWROOT_MNT="/mnt/newroot"

echo ">>> Verwende ROOT_PART=$ROOT_PART"
read -p "Weiter mit ENTER, Abbrechen mit STRG+C..."

sudo mkdir -p "$TOP_MNT" "$NEWROOT_MNT"

### 1. Top-Level (subvolid=5) mounten
echo ">>> Mount Top-Level Btrfs (subvolid=5)..."
sudo mount -o subvolid=5 "$ROOT_PART" "$TOP_MNT"

echo ">>> Inhalt von $TOP_MNT:"
ls "$TOP_MNT"
```

```
### 2. Subvolumes anlegen
echo ">>> Erstelle Subvolumes @, @home, @log, @cache, @snapshots..."
sudo btrfs subvolume create "$TOP_MNT/@"
sudo btrfs subvolume create "$TOP_MNT/@home"
sudo btrfs subvolume create "$TOP_MNT/@log"
sudo btrfs subvolume create "$TOP_MNT/@cache"
sudo btrfs subvolume create "$TOP_MNT/@snapshots"

### 3. Daten in die Subvolumes kopieren
echo ">>> Kopiere Root-Dateisystem nach @ (ohne /home, /var/log, /var/cache, /proc, /sys, /dev, /run, /tmp, /mnt, /media)..."
sudo rsync -aHAX \
  --exclude="/home/*" \
  --exclude="/var/log/*" \
  --exclude="/var/cache/*" \
  --exclude="/proc/*" \
  --exclude="/sys/*" \
  --exclude="/dev/*" \
  --exclude="/run/*" \
  --exclude="/tmp/*" \
  --exclude="/mnt/*" \
  --exclude="/media/*" \
  "$TOP_MNT"/ "$TOP_MNT/@"/

echo ">>> Kopiere /home nach @home..."
sudo rsync -aHAX "$TOP_MNT/home"/ "$TOP_MNT/@home"/ || true

echo ">>> Kopiere /var/log nach @log..."
sudo rsync -aHAX "$TOP_MNT/var/log"/ "$TOP_MNT/@log"/ || true

echo ">>> Kopiere /var/cache nach @cache..."
sudo rsync -aHAX "$TOP_MNT/var/cache"/ "$TOP_MNT/@cache"/ || true

### 4. Alte Verzeichnisse sichern (nicht löschen, nur umbenennen)
echo ">>> Sichere alte Verzeichnisse (home, var/log, var/cache)..."
[ -d "$TOP_MNT/home" ]      && sudo mv "$TOP_MNT/home"      "$TOP_MNT/home.oldroot"      || true
[ -d "$TOP_MNT/var/log" ]   && sudo mv "$TOP_MNT/var/log"   "$TOP_MNT/var.log.oldroot"   || true
[ -d "$TOP_MNT/var/cache" ] && sudo mv "$TOP_MNT/var/cache" "$TOP_MNT/var.cache.oldroot" || true
```

```
### 5. Neue Root-Sicht (subvol=@) mounten
echo ">>> Mount subvol=@ nach $NEWROOT_MNT..."
sudo mount -o subvol=@ "$ROOT_PART" "$NEWROOT_MNT"

### 6. Mountpoints im neuen Root anlegen
echo ">>> Erstelle Mountpoints im neuen Root..."
sudo mkdir -p "$NEWROOT_MNT/home"
sudo mkdir -p "$NEWROOT_MNT/var/log"
sudo mkdir -p "$NEWROOT_MNT/var/cache"
sudo mkdir -p "$NEWROOT_MNT/.snapshots"

### 7. /etc/fstab anpassen
UUID=$(blkid -s UUID -o value "$ROOT_PART")
echo ">>> Gefundene UUID für $ROOT_PART: $UUID"

FSTAB="$NEWROOT_MNT/etc/fstab"
echo ">>> Sicherung von fstab nach fstab.bak-pre-btrfs..."
sudo cp "$FSTAB" "$FSTAB.bak-pre-btrfs"

echo ">>> Passe Root-Eintrag in fstab an (subvol=@)..."
sudo sed -i "s#^\(UUID=$UUID[[:space:]]\+\)/[[:space:]]\+btrfs.*#UUID=$UUID / btrfs
rw,relatime,ssd,space_cache=v2,subvol=@ 0 0#" "$FSTAB"

echo ">>> Füge Subvolume-Mounts für /home, /var/log, /var/cache, /.snapshots hinzu..."
cat <<EOF | sudo tee -a "$FSTAB" >/dev/null

# Btrfs Subvolumes
UUID=$UUID /home       btrfs rw,relatime,ssd,space_cache=v2,subvol=@home      0 0
UUID=$UUID /var/log    btrfs rw,relatime,ssd,space_cache=v2,subvol=@log       0 0
UUID=$UUID /var/cache  btrfs rw,relatime,ssd,space_cache=v2,subvol=@cache     0 0
UUID=$UUID /.snapshots btrfs rw,relatime,ssd,space_cache=v2,subvol=@snapshots 0 0
EOF

echo ">>> fstab nach Anpassung:"
cat "$FSTAB"

### 8. Aufräumen
echo ">>> Unmount..."
sudo umount "$NEWROOT_MNT"
sudo umount "$TOP_MNT"
```

```
echo
echo ">>> FERTIG. Starte jetzt dein installiertes Kubuntu neu."
echo "    Falls etwas schief geht, kannst du im Notfall mit dem Live-System booten"
echo "    und /etc/fstab aus fstab.bak-pre-btrfs wiederherstellen."
```

Benutzung:

1. Nach der Installation wieder mit dem Kubuntu-Live-Stick booten
2. Terminal öffnen, Script z. B. als `kubuntu-btrfs-post.sh` speichern
3. `ROOT_PART` oben anpassen (z. B. `/dev/nvme0n1p2`)
4. Ausführbar machen und laufen lassen:

   ```
   chmod +x kubuntu-btrfs-post.sh
   ./kubuntu-btrfs-post.sh
   ```

5. Reboot von der Platte → Kubuntu sollte jetzt sauber mit:
   - `/` → `@`
   - `/home` → `@home`
   - `/var/log` → `@log`
   - `/var/cache` → `@cache`
   - `/.snapshots` → `@snapshots`

   laufen.

Wenn du willst, können wir das jetzt noch **konkret auf deine Hardware anpassen**:

- Wie heißt deine Zielplatte? (`lsblk`-Ausgabe)
- Planst du Swap (Partition oder Swapfile auf Btrfs)?

Dann schreibe ich dir die beiden Skripte exakt passend (inkl. Swap-Handling) und du musst wirklich nur noch 2× Script starten + 1× Installer klicken.

Revision #2
Created 4 December 2025 12:30:23 by Stefan Mechler
Updated 4 December 2025 12:31:27 by Stefan Mechler