Create a clone drive to provide a backup boot for your Linux machine. This assumes UEFI boot.
# means root user command of course.

1. first create a gpt partition table in the drive that is to hold the OS clone. Be careful not to blow away your original OS or other data drives. For the purposes of this example
sda holds the original OS to be cloned
sdb is the destination drive of the OS clone.
sda and sdb are just example drive names and your installation will likely use different drive names, of the format sdx where x is the drive letter.
We seek to clone the OS from the original OS drive, namely sda to make an OS clone on drive sdb.
For this example, the OS drives are UEFI bootable.

2. Use gparted or parted to create the EFI partition in sdb1 of size between 500MB and 1GB. This efi partition must:
a. be formatted as FAT32
b. have the boot and esp flags set.
c. be of size of about 512MB to say 1GB or so. I used 512MB.

3. Use gparted or parted to create the OS root partition, sdb2 using the remaining space of the OS clone drive sdb. I did not set any flags.
Format the OS root partition, i.e. sdb2, as ext4 (which is what I used).

4. As root, mount the clone drive partition sdb2 to /mnt (for this example - you may use another empty directory in place of /mnt.

```
# mount /dev/sdb2 /mnt
```

5. As root, copy the live OS files from the live OS root directory from the drive to be cloned. In this example, the drive to be cloned is where the live OS resides.

```
# rsync -av --delete --update  --xattrs --acls    --numeric-ids --specials --sparse --devices     --exclude
"/efitest" --exclude "/lost+found" --exclude "/mnt/*" --exclude "/run/*" --exclude "/media/*" --exclude
"/sys/*" --exclude "/tmp/*" --exclude "/dev/*" --exclude "/proc/*"  --exclude "/swapfile" /*  /mnt/
```

if you mounted /dev/sdb2 to a directory other than /mnt, substitute that directory name for /mnt for the instructions here in this how-too. In any case, the mounted directory MUST be excluded, i.e. using --exclude option, from those to be copied above since copying it would result in a recursive copy.

c. Be sure you have the directory /mnt/boot/efi (should be empty) on the new clone drive mounted to /mnt. If not, create it.

also --exclude any other directories you don’t want to copy to the new OS drive. The above are OS directories you MUST exclude e.g. /proc, /swapfile (if it exists), and /mnt (to avoid recursive copies), /run, /tmp, /dev and so on. Add any other directories to the --exclude list that you don’t want replicated on the clone drive. You MUST exclude the above mentioned directories which are symbolically-linked in a live OS otherwise it will cause recursion and problems to copy.
Alternatively, you can copy the OS files from a drive which is not hosting the live OS.

6. Get the UUID numbers for both /dev/sdb1 and /dev/sdb2 using
```
# lsblk
```
or I prefer gnome-disks. gnome-disks will show the UUIDs for the selected partition
Edit /mnt/etc/fstab and update the lines:
```
UUID=UUIDnumber of the clone disk’s partition 2 /            ext4    errors=remount-ro 0
UUID=UUIDnumber of the clone disk’s partition 1 /boot/efi      vfat    umask=0077      0       1
```
This is important otherwise the OS will boot into emergency mode only (read only).

7. Mount the efi partition to /mnt/boot/efi so that GRUB can write to the efi partition in the right location. Note that an empty /mnt/boot/efi directory MUST exist to allow this.
command:
```
# mount /dev/sdb1 /mnt/boot/efi
```
Now it’s time to prepare the chroot to allow proper installation of GRUB for the efi boot.

8. Bind the directories of /mnt/ using the following commands as root (note that # is the root prompt and not something you type in as part of the command)
Note that wee should have the sdb2 partition mounted at /mnt and the sdb1 partition mounted at /mnt/boot/efi prior to executing the following commands.
command are all at the root prompt #:
```
# mount -B /dev /mnt/dev
# mount -B /dev/pts /mnt/dev/pts
# mount -B /sys /mnt/sys
# mount -B /proc /mnt/proc
```
get into the chroot
```
# chroot /mnt
# mount -t efivarfs none /sys/firmware/efi/efivars
# update-grub
# grub-install --target x86_64-efi --efi-directory /boot/efi --boot-directory /boot
# update-grub
# update-initramfs -u
```
leave chroot
```
#exit
```

you then can and should umount the binds
but I just reboot the machine to accomplish that.

9. shutdown your machine and pull out he original OS drive or get into your BIOS and set the clone drive as your boot drive. Even if you disconnect your original OS drive, you might have to set your BIOS to boot from the clone drive depending on your system.
The above should work and was successfully tried on Ubuntu 22.04 (hope I didn’t leave out any steps). A similar procedure *might* work to install your OS to a root ZFS mirror, but that’s for another day.