Clone Linux System

Overview

This page assumes that system is installed in EFI mode having: a 200-500MB vfat/fat32 “EFI system” partition.

Get Source Images

Create a live USB and boot system from USB. Once booted into the live-cd, mount the source filesystem. First check the correct device with lsblk

```
# lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0 931,5G  0 disk
├─sda1   8:1    0   512M  0 part /boot/efi
├─sda2   8:2    0  18,6G  0 part /
└─sdb3   8:3    0   9,8G  0 part [SWAP]
sdb      8:0    0 931,5G  0 disk
├─sdb1   8:1    0   512M  0 part /boot/efi
├─sdb2   8:2    0  18,6G  0 part /
├─sdb3   8:3    0   9,8G  0 part [SWAP]
└─sdb4   8:4    0 902,6G  0 part /media/storage
```

We need to copy the **“/boot/efi”** and **“/“** partitions to the new system.

Get EFI Partition

We will use dd to get an exact image copy and the size of source and target partitions are usually similar. Create Backup Image:

```
# dd bs=4M if=/dev/sdb1 | gzip > efi-image.gz
# sync
```

Get Root Partition

For the root partition we only want to copy the files. This is faster since the root partitions is usually partially filled.

```
# mkdir /tmp/source_sdXY
# mount -t ext4 /dev/sdXY /tmp/source_sdXY
```

```
# tar -zcvf image-sdXY.tgz /tmp/source_sdXY
```

Create Targte Disk

Again, first check with lsblk the disk structure, before proceeding:

```
# lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0 931,5G  0 disk
├─sda1   8:1    0   512M  0 part /boot/efi
├─sda2   8:2    0  18,6G  0 part /
└─sdb3   8:3    0   9,8G  0 part [SWAP]
sdb      8:0    0 931,5G  0 disk
├─sdb1   8:1    0   512M  0 part /boot/efi
├─sdb2   8:2    0  18,6G  0 part /
├─sdb3   8:3    0   9,8G  0 part [SWAP]
└─sdb4   8:4    0 902,6G  0 part /media/storage
```

Write EFI Partition

```
# gzip -dc efi-image.gz | dd bs=4M of=/dev/sdbX
# sync
<code>
==== Write Root Partition ====
Empty and create a fresh filesystem on the root partition of the target
disk:
  # mkfs.ext4 -l root /dev/sdbX
Mount the root partition. Use the same /tmp/source_sdXY directory as we used
for the tar.
  # mount -t ext4 /dev/sdX /tmp/source_sdXY
Now restore the files to the partitions.
  # cd /
  # tar -zxvf image-sdX1.tgz

===== install a bootloader =====
At this point we have all the files we need on the new system, but we need
to make the new system bootable. We will chroot into the newly extracted
filesystem to install a bootloader.

First, change directories to where the new installation is mounted (e.g.
/tmp/source_sdXY in the example above):

  # cd /tmp/source_sdXY

Then we'll use mount ---bind to give the chroot access to the linux special
directories. The bind option of the mount command allows you to remount part
of a file hierarchy at a different location while it is still available at
```

```
the original location. The format of this command is as follows.
<code>
# for i in /dev /dev/pts /proc /sys /run; do mount --bind $i .$i; done
Which is similar to:
# mount --bind /dev /source_sdXY/dev
# mount --bind /dev/pts /source_sdXY/dev/pts
# mount --bind /proc /source_sdXY/proc
# mount --bind /sys /source_sdXY/sys
```

Because we installing in EFI mode we also need to give our chroot access to the EFI partition we mounted earlier. mount –bind comes to the rescue again here, we simply bind mount the livecd mount point into the /boot/efi directory inside the chroot (/boot/efi is where grub expects to find the EFI partition). First mount the new EFI partition:

```
# mkdir /tmp/efi-part
# mount /dev/sdX1 /tmp/efi-part
```

Now bind it in the new croot partition:

```
# cd /tmp/source_sdXY
# mkdir -p boot/efi
# mount --bind /tmp/efi-part boot/efi
```

Now that we have access to the Linux special folders (and the EFI partition), we can use the chroot command to actually use our source installation:

```
# chroot /tmp/source_sdXY
```

At this point you should have a shell inside the same Linux environment you originally copied. We can run grub-install from inside the chroot to update the boot partition.

Install bootloader

Run grub-install against the drive you installed to. In my case that’s /dev/sdb, but this may be different on your machine. Next we install grub to our drive, thereby making it bootable. Be careful to install grub to a drive and not to a partition.

```
# grub-install /dev/sdb
# update-grub
```

If all went well you will see messages saying that grub was successfully installed. When you see this feel free to reboot and check out your freshly cloned installation.

Troubleshooting

If you get a a warning saying that “EFI variables cannot be set on this system,” you may need to mount the EFI vars into the chroot (h/t Jesse Dhillon):

```
# mount -t efivarfs none /sys/firmware/efi/efivars
```

If you get error messages when installing in EFI mode it's possible grub's autodetect got confused and tried to use MBR mode when it should've used EFI. You may be able to succesfully perform an EFI install by forcing EFI mode like so:

```
@ grub-install --target=x86_64-efi
```

Write Target System

Create a live USB and boot system from USB. Once booted into the live-cd, mount your destination filesystem.

Mount the partition your broken Linux installation is on. If you are not sure which it is, launch GParted (included in the Live CD) and find out. It is usually a EXT4 Partition. Replace the XY with the drive letter, and partition number, for example: sudo mount /dev/sda1 /mnt.

```
# mount /dev/sdXY /mnt
```

Now bind the directories that grub needs access to to detect other operating systems, like so.

```
# mount --bind /dev /mnt/dev
# mount --bind /dev/pts /mnt/dev/pts
# mount --bind /proc /mnt/proc
# mount --bind /sys /mnt/sys
```

Internet access For internet access inside chroot:

```
# mv /mnt/etc/resolv.conf /mnt/etc/resolv.conf.org
# cp /etc/resolv.conf /mnt/etc/resolv.conf
```

Now we jump into that using chroot.

```
# chroot /mnt
```

Now install, check, and update grub. This time you only need to add the drive letter (usually a) to replace X, for example: grub-install /dev/sda, grub-install –recheck /dev/sda.

```
# grub-install /dev/sdX
# grub-install --recheck /dev/sdX
```

Alternatively, in case of persistent problems, you can purge and reinstall grub2, make new config files:

```
apt-get remove --purge grub-pc grub-common
apt-get install grub-pc
grub-mkconfig
```

```
update-grub
grub-install /dev/sda
```

Now grub is back, all that is left is to exit the chrooted system and unmount everything:

```
# exit
# umount /mnt/sys
# umount /mnt/proc
# umount /mnt/dev/pt
# umount /mnt/dev
# umount /mnt
```

Shut down and turn your computer back on, and you will be met with the default Grub2 screen.