How to move Linux root partition to another drive quickly

Dominik Gacek · Follow

4 min read · Jun 21, 2019

There's a bunch of information over internet on how to clone the Linux drives or partitions between other drives and partitions using solution like `partclone`, `clonezilla`, `partimage`, `dd` or similar, and while most of them are working just fine, they're not always the fastest possible way to achieve the result.

Today I want to show you another approach that combines most of them, and I am finding it the easiest and fastest of all.

Assumptions:

1. You are using GRUB 2 as a boot loader
2. You have two disks/partitions where a destination one is at least the same size or larger than the original one.

Let's dive in into action.

Just "dd" it…

First thing that we have to do, is to create a direct copy of our current root partition from our source disk into our target one.

Before you start, you have to know what are the device names of your drives, to check on that type in:

```
sudo fdisk -l
```

You should see the list of all the disks and partitions inside your system, along with the corresponding device names, most probably something like `/dev/sdx` where the `x` will be replaced with proper device letter, in addition to that you'll see all of the partitions for that device prefixed with partition number, so something like `/dev/sdx1`

Based on the partition size, device identifier and the file-system, you can say what partitions you'll switch your installation from and which one will be the target one.

I am assuming here, that you already have the proper destination partition created, but if you do not, you can utilize one of the tools like GParted or similar to create it.

Once you'll have those identifiers, let's use `dd` to create a clone, with command similar to.

```
sudo dd if=/dev/sdx1 of=/dev/sdy1 bs=64K conv=noerror,sync
```

Where `/dev/sdx1` is your source partition, and `/dev/sdy1` is your destination one.

It's really important to provide the proper devices into `if` and `of` arguments, cause otherwise you can overwrite your source disk instead!

The above process will take a while and once it's finished you should already be able to mount your new partition into the system by using two commands:

```
sudo mkdir /mnt/new
sudo mount /dev/sdy1 /mnt/new
```

There's also a chance that your device will be mounted automatically but that varies on a Linux distro of choice.

Once you execute it, if everything went smoothly you should be able to run

```
ls -l /mnt/new
```

And as the outcome you should see all the files from the core partition, being stored in the new location.

It finishes the first and most important part of the operation.

Now the tricky part

We do have our new partition moved into shiny new drive, but the problem that we have, is the fact that since they're the direct clones both of the devices will have the same UUIDs and if we want to load your installation from the new device properly, we'll have to adjust that as well.

First, execute following command to see the current disk uuid's

```
blkid
```

You'll see all of the partitions with the corresponding UUID.
Now, if we want to change it we have to first generate a new one using:

```
uuidgen
```

which will generate a brand new UUID for us, then let's copy it result and execute command similar to:

```
sudo tune2fs /dev/sdy1 -U cd6ecfb1-05e0-4dd7-89e7-8e78dad1fa0e
```

where in place of `/dev/sdy1` you should provide your target partition device identifier, and in place of `-U` flag value, you should paste the value generated from `uuidgen` command.

Now the last thing to do, is to update our fstab file on new partition so that it'll contain the proper UUID, to do this, let's edit it with.

```
sudo vim /etc/fstab
# or nano or whatever editor of choice
```

you'll see something similar to the code below inside:

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name
devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sdc1 during installation
UUID=cd6ecfb1-05e0-4dd7-89e7-8e78dad1fa0e / ext4 errors=remount-ro 0 1
# /home was on /dev/sdc2 during installation
UUID=667f98f4-9db1-415b-b326-65d16c528e29 /home ext4 defaults 0 2
/swapfile none swap sw 0 0
UUID=7AA7-10F1 /boot/efi vfat defaults 0 1
```

The bold part is important for us, so what we want to do, is to paste our new UUID replacing the current one specified for the `/` path.

And that's almost it

The last part you have to do is to simply update the grub.

There are a number of options here, for the brave ones you can edit the `/boot/grub/grub.cfg`

Another option is to simply reinstall grub into our new drive with command:

```
sudo grub-install /dev/sdx
```

And if you do not want to bother with editing or reinstalling grub manually, you can simply use the tool called `grub-customizer` to have a simple and easy GUI for

all of those operations.

Happy partitioning! :)