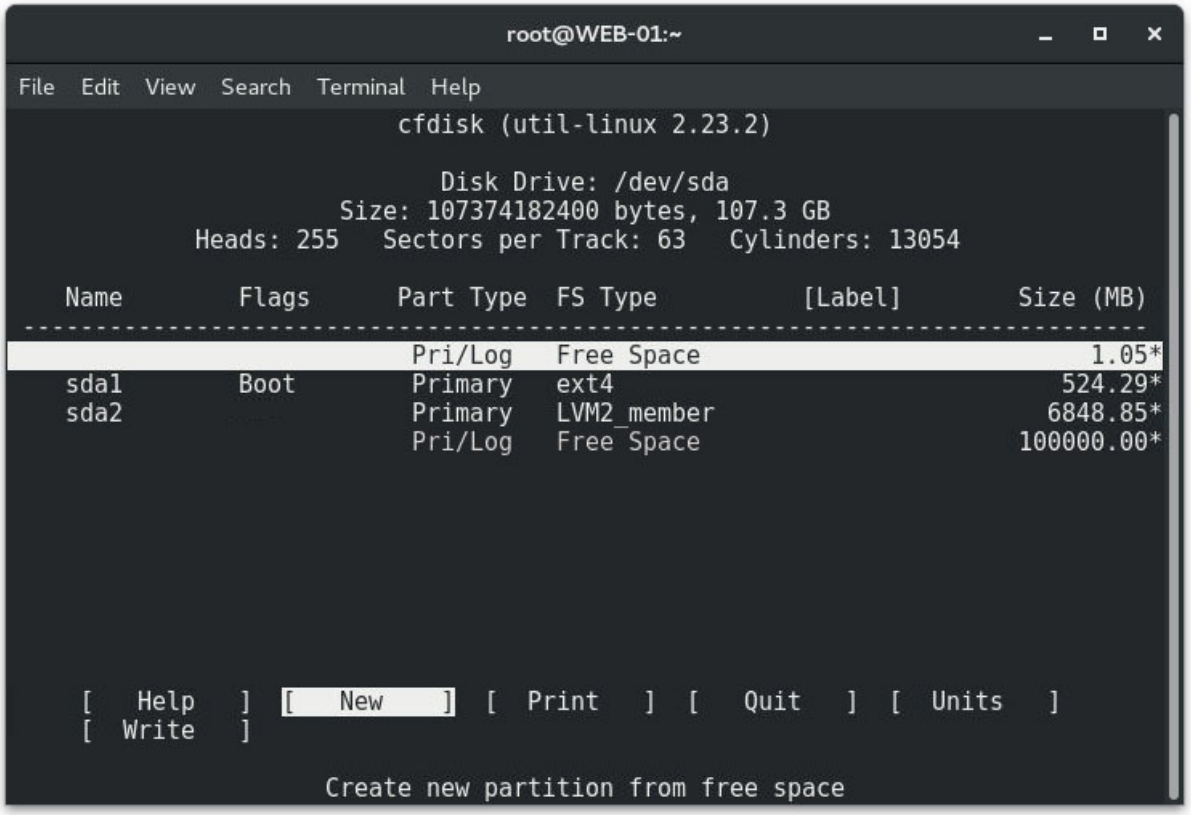
Resize-Extend a disk partition with unallocated disk space in Linux

Expanding disk partitions to use all the available (unallocated) disk space is a common issue among Linux Administrators, especially when working in a VMware-based Cloud environment: deploying a Linux VM from an existing template will often lead to disk partitions smaller than the disk space allocated during the VM configuration phase.

Consider the following scenario:



This screen can be obtained by running **cfdisk** from the terminal and shows the following:

a 524MB boot partition [sda1]
a 6.8GB drive [sda2], used by the Linux OS and all its installed packages.
100GB of unallocated space
It would be great to extend that puny 6.8GB partition and make it become a 106.8GB drive… How can we do that? If you take a look

around you'll see that the web is awfully short of a quick and effective tutorial for this: that's why I eventually chose to write my own guide: here it is!

Luckily enough, we won't need anything fancy to perform our task: we're just going to make good use of fdisk , pvresize , lvdisplay and lvextend , some handy command-line tools shipped with any Linux distribution: that's great to hear, since it means that this tutorial will work for any Linux distro, including CentOS 5.x, CentOS 6.x, CentOS 7.x, RHEL, Ubuntu, Debian and more!

Step 1: Alter the Partition Table

The first thing we need to do is to modify our partition table to make sda2 end at end of disk: don't worry, you will not lose your existing data! However, this task will require a reboot in order to write the changes that we're going to make and also to re-read the updated partition table.

Let's start with running the following command:

fdisk /dev/sda

This will make the terminal enter in fdisk mode:

- once there, type **p** to print the current partition table: it's very important to keep note of the numeric values of the START and END columns for the /**dev/sda2** partition, as we're going to need them soon enough. If you want to be sure to not lose them or typing them wrong, just print-screen or paper-print them.
- Once done, type **d** to access the delete mode, and then the number of the partition record that you want to remove (that would be **2** in the above scenario). Again, DO NOT WORRY: you're not deleting the partition data, just its mapping addresses on the partition table.
- Right after that, type n to create a brand-new second partition: choose the same partition mode of the previous one (that would be **Primary** in the above scenario), then input the START numeric value you've recorded earlier – which should be the suggested default; also, make sure that the end is at the end of the disk – which should also be the default.

- Last but not least, we need to change the partition type from Linux to Linux LVM: to do so, type **t** to access the change partition type mode, then **2** , then **8e** and that's it.
- When done, type **p** to review your new partition layout. Be sure to triple-check that the start of the new second partition is exactly where the old second partition was: this is very important! In case you did it wrong, type **d** to delete the wrong partition and start over.
- If everything looks right, issue **w** to write the partition table to disk.

Step 2: Reboot

Right after writing the new partition table to disk, you'll immediately get a system error message due to the fact that the partition table couldn't be accessed for read, because the disk is in use. That's why we need to reboot our system.

Step 3: Expand the LVM Partition

Use the resize2fs utility to extend the EXT3/4 filesystem to utilise the additional space in the partition e.g.:

resize2fs /dev/sdb1

Note: when running resize2fs, if no size is specified, the filesystem will be extended to utilise all available/remaining space in the partition.