

Ubuntu LVM Guide - Part 1

This guide will help you get to grips with the fundamental concepts of logical volume management in Ubuntu. We'll then run through preparing disks for lvm, extending and reducing an existing logical volume, and we'll show you how to add a new logical volume.

In this guide, we'll start with a **default Ubuntu 12.10 (quantal quetzal) lvm installation** and run you through how you can use lvm as a flexible solution for your storage needs. Having said that, any other Ubuntu release or other Linux based distribution using the same set of tools (called lvm2) will

find this stuff useful.

Note that in this guide, we'll carry out all the lvm actions in our examples via the command line tools whilst using a gui tool to help illustrate what has been done.

It's also worth pointing out now that the Ubuntu 12.10 installer does not support creating the partitions for a custom lvm layout itself, but it will allow you to use a custom layout that you have already prepared via live cd/usb (in which case you would select the "something else" option rather than "Use LVM with the new Ubuntu installation". We shall cover this in a future guide).

The default layout should be OK for most people's purposes, in any case bear in mind that you can always change the layout yourself later if required (which we shall also be covering in a future guide).

What Is LVM?

In human terms, the logical volume manager (LVM) in Ubuntu & other Linux distributions is software that allows for a higher level and more flexible

management of your hard disks by creating logical ones (called logical volumes).

These logical volumes can be made up from different physical hard disks, yet thanks to software mapping they will appear as one volume on which a file system can be installed.

In other words, small physical chunks of disk space from anywhere on any disk can be melded together to provide what appears to be one big disk. Through LVM, you can create what appears to be a normal usable partition that underneath, potentially consists of different physical underlying disks and/or partitions.

As you'll see later in this guide (and also in future follow up guides), lvm allows you to do some really neat stuff.

Always Backup Important Data Before Modifying Volumes!

Before you carry out any sort of operations on your disks (whether using lvm or not) you should always back up important data. There is always a

risk of accidental data loss.

Why Would One Want To Use LVM?

There are many reasons why you may want to use lvm. Although there is a bit of a learning curve to using lvm, once you know how to use it, it allows for far more flexibility than messing around with physical partitions ever will. Below are some of the reasons people choose to use lvm:

1. **No need to know how much space a partition will need in advance because logical volumes can be easily be resized.** You can even extend them while they are in use! If you run out of space, just throw a new physical disk or partition into your setup then allocate that resource to wherever you need it. Alternatively you could reduce the size of one of your volumes and give the resource to another volume that's running low on space.
2. **You can move a file system on a logical volume to another underlying physical disk while still in use.** This allows you to do things like remove a hard disk from your system without any down time.
3. **You can create snapshots of your current system which you can easily revert to** (for example if you screwed something up by installing new software or messed up a config file somewhere.). You can mount these snapshots if need be, you can even boot into a snapshot if required. One nice thing about snapshots is that they don't use up a lot of space. It's only when data is changed on your

main volume, that the original data gets copied to the snapshot (this concept is called copy on write).

4. You can create one or more mirrors of a logical volume on different underlying disks so that if your main hard disk fails, you'll have no data loss.
5. You can create striped logical volumes which can improve performance because the underlying disks can be read in parallel.

The Main Concepts

In order to be able to use a physical disk or physical partition with lvm, it must be initialised for lvm use.

As part of this stage, physical disks/partitions get carved up (conceptually, by the lvm software - not physically!) into small chunks (of 4MiB by default) called **physical extents**.

After this initialisation, the relevant physical disk or physical partition is referred to as a **physical volume**. Many different underlying disks can be modelled this way. The space they provide can then be thrown into the same pot, or to use the correct term **volume group**. In one of these volume groups, lvm can use these physical extents as the building blocks for a **logical volume**, which appears to your Linux system as a normal partition

on a normal hard disk. As such, the logical volume is where your file system gets created.

In the logical volume, the chunks that map to physical extents are called **logical extents**.

So to look at it another way, the contiguous logical extents which make up the logical volume can map to non-contiguous physical extents, thus residing in different physical locations on a disk and even different disks. You can create more than one logical volume from the pool of physical extents in a given volume group.

A volume group is responsible for tying together a pool of physical disk space to one or more usable logical volumes.

If need be, you can create more than one volume group.

Most desktop users will probably only need to use one lvm volume group. However, in special cases (e.g. production servers), people may want to create more than one volume group. For example for a server that streams video the owner may want at least two volume groups, one consisting of several normal physical disks perhaps used for logging, and another volume group consisting of super fast disks to fetch the data for streaming.

Ubuntu 12.10 Default LVM Layout

For illustration purposes, let's just show you what a **default 12.10 lvm install** onto one 500GB hard disk would look like.

The default install consists of 3 partitions: `/dev/sda1` (used for `/boot`), `/dev/sda2` which is used for an "extended partition" which contains `/dev/sda5`. This `/dev/sda5` is a partition configured for lvm. It is used to provide space for the two default logical volumes. One for swap space, and the other for the root file system `/`.

How does this appear to `fdisk` and `gparted`?

`fdisk`

The command line tool `fdisk` sees this:

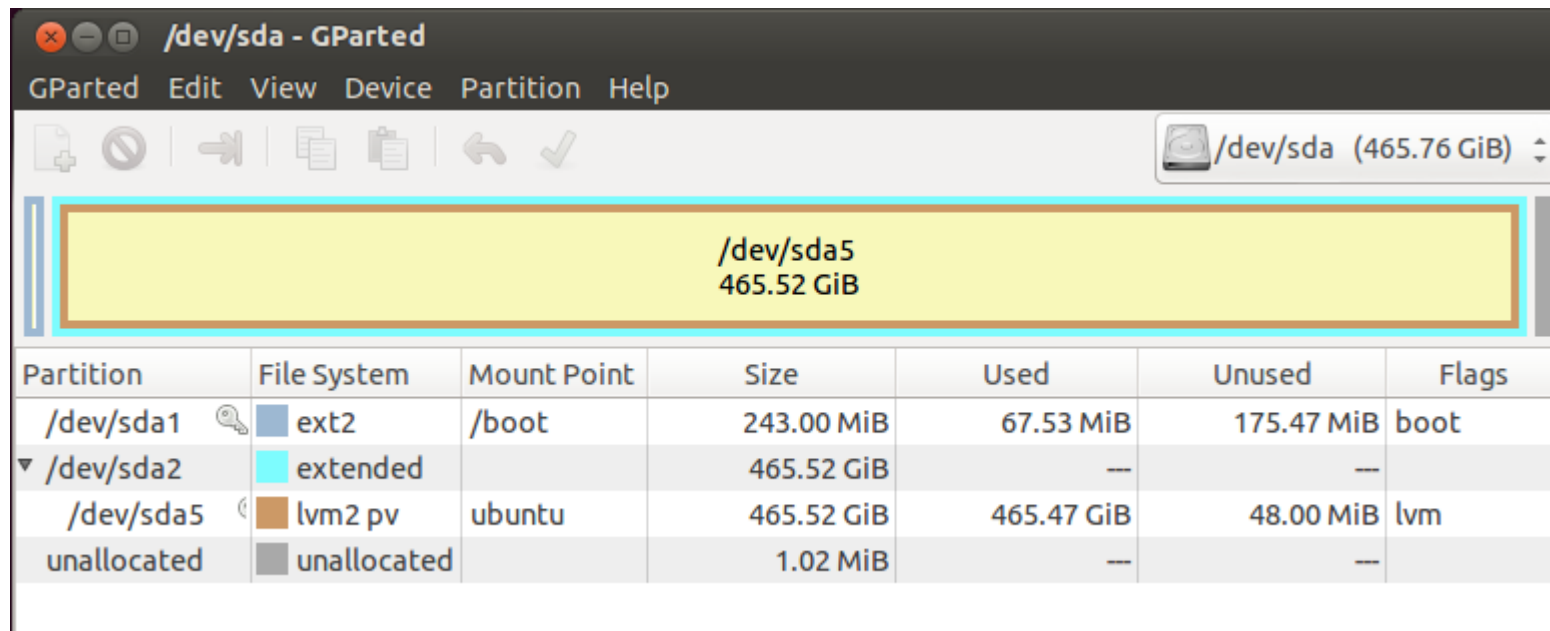
```
tutonics@andromeda: ~  
tutonics@andromeda:~$ sudo fdisk -l /dev/sda  
  
Disk /dev/sda: 500.1 GB, 500107862016 bytes  
255 heads, 63 sectors/track, 60801 cylinders, total 976773168 sectors  
Units = sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disk identifier: 0x000226ca  
  
   Device Boot      Start         End      Blocks   Id  System  
/dev/sda1  *        2048       499711       248832   83   Linux  
/dev/sda2                501758     976771071     488134657    5   Extended  
/dev/sda5                501760     976771071     488134656   8e   Linux LVM  
tutonics@andromeda:~$
```

As you can see, the partition `/dev/sda5` is an lvm partition. The partition id for it is set as type 8e.

It is recommended (good practice) that any partition used by lvm is marked as type 8e.

gparted

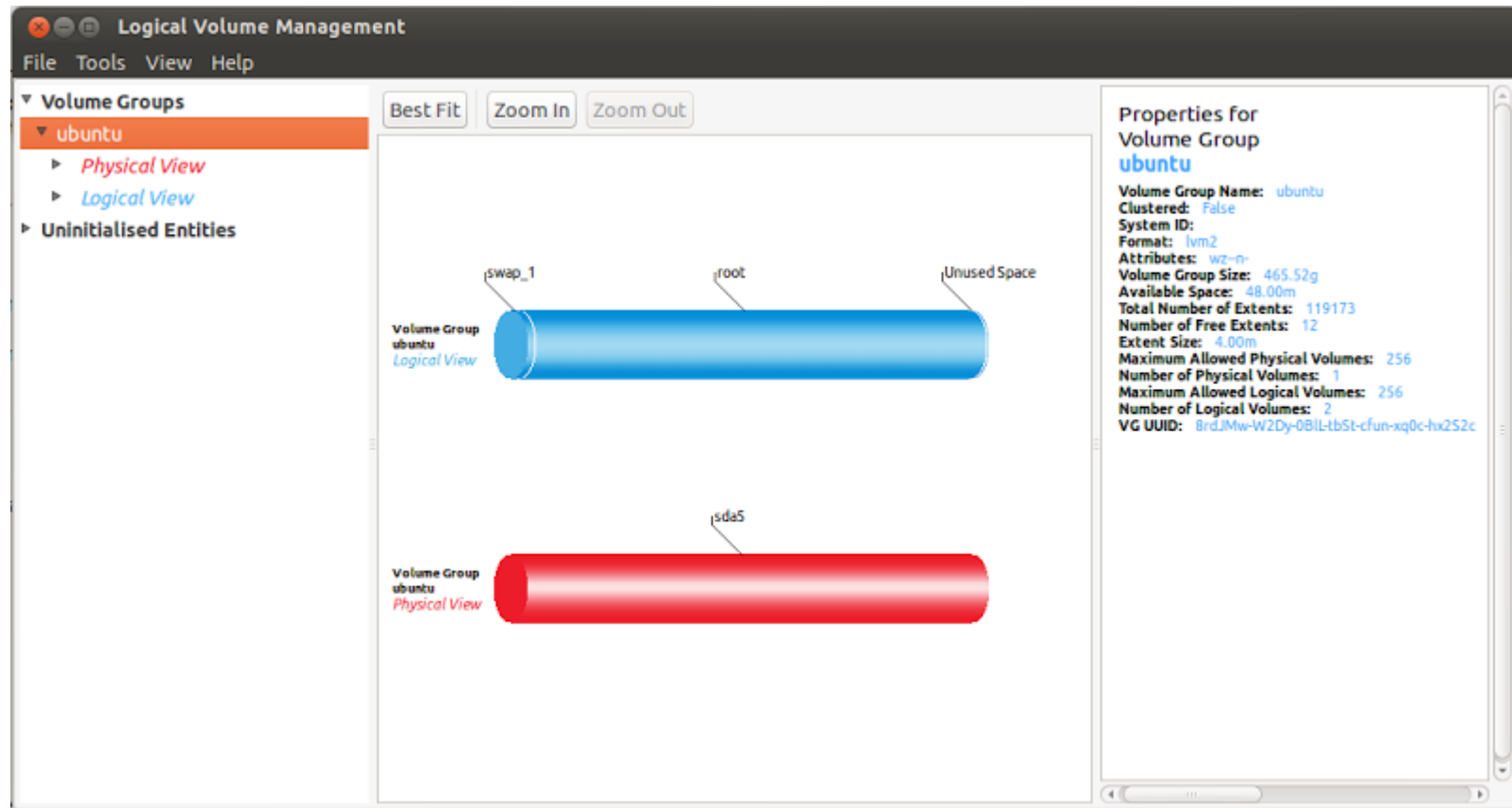
When we look at the same default layout using gparted (shown below), we get a better idea about how things are laid out physically on the hard disk:



We can see that /dev/sda2 is indeed an extended partition and it holds a partition used for an lvm physical volume (pv). But still we have no information about the lvm's logical volume layout or the way that physical volumes map to the logical volumes.

As you may have guessed from the above fdisk and gparted screenshots, these tools do not yet have the functionality to identify any logical volumes. They can only identify that a given partition is being used by the logical volume manager.

Below is a screenshot from a handy graphical tool which you can use to view all logical and physical volumes.



As you can see, the blue cylinder shows the logical volumes `swap_1` and `root` which are installed as the default 12.10 lvm layout. The red cylinder shows the underlying physical layout, which consists of one single partition called `/dev/sda5`. As you can see from the text on the right of the screenshot, there is one volume group called **ubuntu**. By

clicking on the volumes under "physical view" and "logical view", you can see more information about each physical and logical volumes respectively. In this case, with a default install you'll be able to see details about your swap_1 and root logical volumes.

The above tool (which can be quite handy) is called **system-config-lvm**. To install it if you're using a laptop or desktop:

```
sudo apt-get install system-config-lvm
```

You can also use this tool to perform your lvm actions for you. However, it does not recognise raid properly, so could mislead people using raid. To properly understand how to administer lvm, you really ought to know how to use the command line tools (which are packaged as lvm2 and already included in a Ubuntu 12.10 lvm install). Having said that you can get most jobs done using the system-config-lvm gui.

An interesting thing to point out about the default Ubuntu 12.10 layout is that /dev/sda1 (where /boot got installed) is on a real physical partition. Strictly speaking, /boot could also have been installed on a logical volume

rather than a real physical partition because Ubuntu uses grub2 which does recognise logical volumes.

This is a good time to point out where you can find the logical volumes on your filesystem (or rather what to call them when you need to reference them).

You can use this type of path: `/dev/volumeGroup/logicalVolume` or this type of path: `/dev/mapper/volumeGroup-logicalVolume`

So, taking `swap_1` as an example, you can use either of these two ways to reference it:

```
/dev/ubuntu/swap_1
```

or

```
/dev/mapper/ubuntu-swap_1
```

If you do an `"ls -la"` on these, you'll see they both point to the same underlying block device.

Now then, lets press on using the default 12.10 lvm layout.

LVM Display Commands

Lets start out at the command line by introducing the three main display commands: `pvdisplay`, `vgdisplay`, and `lvdisplay`.

To see what physical volumes are configured for lvm, use the **`pvdisplay`** command. When used with its `"-m"` option, you can also see the range of physical extents and how they map to logical extents.

```
sudo pvdisplay -m
```

Output will look something like that shown below:

```
tutonics@andromeda: ~  
tutonics@andromeda:~$ sudo pvdisplay -m  
--- Physical volume ---  
PV Name                /dev/sda5  
VG Name                ubuntu  
PV Size                465.52 GiB / not usable 2.00 MiB  
Allocatable           yes  
PE Size                4.00 MiB  
Total PE               119173  
Free PE               12  
Allocated PE          119161  
PV UUID               ALgG26-romZ-Fz0A-QEDr-HK5C-6USZ-oTZe1M  
  
--- Physical Segments ---  
Physical extent 0 to 117145:  
  Logical volume      /dev/ubuntu/root  
  Logical extents     0 to 117145  
Physical extent 117146 to 119160:  
  Logical volume      /dev/ubuntu/swap_1  
  Logical extents     0 to 2014  
Physical extent 119161 to 119172:  
  FREE  
  
tutonics@andromeda:~$
```

Note that if you only want to display info about specific volume(s), just include the name(s) as part of the command, so for example to see info about /dev/sda5 specifically, you would use:

```
sudo pvdisplay -m /dev/sda5
```

As we described earlier, a volume group manages a pool of physical extents and the logical volumes you've built from them. To see details about the volume groups that exist on your system, use the `vgdisplay` command:

```
sudo vgdisplay
```

This will produce output like that shown below:

```
tutonics@andromeda: ~  
tutonics@andromeda:~$ sudo vgdisplay  
--- Volume group ---  
VG Name                ubuntu  
System ID                 
Format                 lvm2  
Metadata Areas          1  
Metadata Sequence No   3  
VG Access               read/write  
VG Status               resizable  
MAX LV                  0  
Cur LV                  2  
Open LV                  2  
Max PV                  0  
Cur PV                  1  
Act PV                  1  
VG Size                 465.52 GiB  
PE Size                 4.00 MiB  
Total PE                119173  
Alloc PE / Size         119161 / 465.47 GiB  
Free PE / Size           12 / 48.00 MiB  
VG UUID                 8rdJMw-W2Dy-0BLL-tbSt-cfun-xq0c-hx2S2c  
  
tutonics@andromeda:~$
```

To see a lot more information (verbose mode), use with "-vv".
Note that to only see the info about a specific volume group or groups, just include the name(s) as part of the command, e.g to display the ubuntu volume group, use:

```
sudo vgdisplay ubuntu
```


Finally, if you'd like to display the logical volumes on your system, use the **lvdisplay** command.

```
sudo lvdisplay
```

Sample output is shown below:

```
tutonics@andromeda: ~  
tutonics@andromeda:~$ sudo lvsdisplay  
--- Logical volume ---  
LV Path                /dev/ubuntu/root  
LV Name                 root  
VG Name                 ubuntu  
LV UUID                 CGPN6p-But0-3gqv-fn0q-T1EY-eMg0-tNa4Wj  
LV Write Access         read/write  
LV Creation host, time ubuntu, 2012-11-18 15:10:56 +0000  
LV Status                available  
# open                  1  
LV Size                 457.60 GiB  
Current LE              117146  
Segments                1  
Allocation               inherit  
Read ahead sectors      auto  
- currently set to      256  
Block device            252:0  
  
--- Logical volume ---  
LV Path                /dev/ubuntu/swap_1  
LV Name                 swap_1  
VG Name                 ubuntu  
LV UUID                 pttfRm-UBbh-YrWt-hL7u-W1aK-heZN-KecwPH  
LV Write Access         read/write  
LV Creation host, time ubuntu, 2012-11-18 15:10:57 +0000  
LV Status                available  
# open                  2  
LV Size                 7.87 GiB  
Current LE              2015  
Segments                1  
Allocation               inherit  
Read ahead sectors      auto  
- currently set to      256  
Block device            252:1  
  
tutonics@andromeda:~$
```

As with the other display commands, to display details about specific logical volume(s), just include the name(s) as part of the command, so to display info about swap_1 logical volume, you could use:

```
sudo lvdisplay /dev/ubuntu/swap_1
```

Preparing New Disks & Partitions For LVM Usage

You have two choices when choosing how to add physical disk space to logical volume management. You can either use a **whole disk** or you can use one or more **partitions** on a disk.

When other operating systems have visibility of the disk, it is **recommended that you pass partitions to lvm rather than whole disks**. The reason for this is because if a whole disk is used for lvm, an operating system other than linux may not recognise the lvm metadata and show the disk as being empty, thus adding risk of accidental overwrite.

Prepare A Whole Disk For LVM

Note when you setup a whole disk for use with lvm, **any existing data on that disk will be lost!** For whole disks (i.e. when you want to use the whole disk for lvm) you first need to erase any existing partition table by zeroing out the first sector. Suppose we added a new disk which was represented by the device `/dev/sdb`, we'd use:

```
dd if=/dev/zero of=/dev/sdb bs=512 count=1
```

Now the whole disk `/dev/sdb` is ready to be initialised for use with lvm.

Prepare A Partition For Use With LVM

Alternatively, to prepare a partition for lvm, when you create the partition, you should follow the below instructions for setting the lvm type.

Using fdisk

1

Run the following command:

```
sudo fdisk /dev/sdb
```

Please verify that this is the disk you wish to use, press "p" as shown below.

```
tutonics@andromeda:~$ sudo fdisk /dev/sdb

Command (m for help): p

Disk /dev/sdb: 250.1 GB, 250059350016 bytes
255 heads, 63 sectors/track, 30401 cylinders, total 488397168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0002ff66

   Device Boot      Start         End      Blocks   Id  System
Command (m for help):
```

If you are certain that this is the disk you want to use, then proceed. As you can see, in our example we're using a new 250GB disk.

2

Press "n" to create a new partition, then press "p" to indicate that its a primary partition.

```
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1):
```

3

Now you need to enter the partition number. If this is a new disk, press "1" here because we're creating the first partition, otherwise press the appropriate partition number for your new partition.

```
Partition number (1-4, default 1): 1
First sector (2048-488397167, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-488397167, default 488397167):
Using default value 488397167
Command (m for help):
```

As you can see above, you'll be prompted to enter the first sector and the last sector to use for the partition. If you're using the whole disk for LVM (as we are), just hit return to accept the default first, then hit return again to accept the default last sector.

4

Next, we need to set the type of the partition to 8e (which signifies LVM).

Hit **t**, then enter **8e** as shown below.

```
Command (m for help): t
Selected partition 1
Hex code (type L to list codes): 8e
Changed system type of partition 1 to 8e (Linux LVM)
Command (m for help):
```

5

Now we need to write those changes by entering "w":

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
tutonics@andromeda:~$
```

6

You can now verify that these changes have been made by running:

```
sudo fdisk -l /dev/sdb
```

You can see the output from our `/dev/sdb` shown below which shows that `/dev/sdb1` has system type as "Linux LVM":

```
tutonics@andromeda:~$ sudo fdisk -l /dev/sdb

Disk /dev/sdb: 250.1 GB, 250059350016 bytes
81 heads, 63 sectors/track, 95707 cylinders, total 488397168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0002fffc6

   Device Boot      Start         End      Blocks    Id  System
/dev/sdb1             2048     488397167     244197560    8e  Linux LVM
tutonics@andromeda:~$
```


The partition `/dev/sdb1` is now ready to be initialised for Logical Volume Management.

Creating A Physical Volume From Physical Disks/Partitions

Assuming you've already prepared your disk or partition for lvm as described above, before adding it to a volume group's pool, the physical disk or partition must be "initialised" to create a physical volume that is recognised by lvm.

In order to initialise a disk or partition for use with lvm, you need to use the **pvcreate** command.

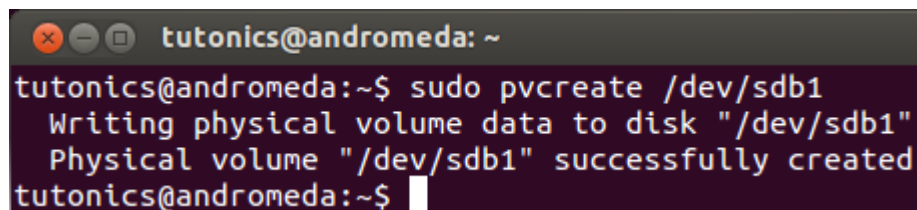
To initialise a whole disk as a physical volume, use:

```
sudo pvcreate /dev/sdb
```

To initialise a partition as physical volume, just use the partitions device name. For example for the first partition on disk /dev/sdb (which is called /dev/sdb1), use:

```
sudo pvcreate /dev/sdb1
```

The output from using pvcreate to initialise /dev/sdb1 as an lvm physical volume is shown below:

A terminal window with a dark background and light text. The prompt is 'tutonics@andromeda: ~'. The command 'sudo pvcreate /dev/sdb1' has been entered. The output shows 'Writing physical volume data to disk "/dev/sdb1"' and 'Physical volume "/dev/sdb1" successfully created'. The prompt returns to 'tutonics@andromeda:~\$' with a cursor.

```
tutonics@andromeda: ~  
tutonics@andromeda:~$ sudo pvcreate /dev/sdb1  
Writing physical volume data to disk "/dev/sdb1"  
Physical volume "/dev/sdb1" successfully created  
tutonics@andromeda:~$
```

Adding Physical Volumes To A Volume Group

In order to add a physical volume to a volume group's pool of physical space, you need to use the **vgextend** command.

In our example, we'll add the partition /dev/sdb1 (approx 250GB) to the

ubuntu volume group.

Here's a screenshot of the **ubuntu** volume group's info before we do that:

```
tutonics@andromeda: ~  
tutonics@andromeda:~$ sudo vgdisplay ubuntu  
--- Volume group ---  
VG Name                ubuntu  
System ID  
Format                 lvm2  
Metadata Areas         1  
Metadata Sequence No   3  
VG Access               read/write  
VG Status               resizable  
MAX LV                 0  
Cur LV                2  
Open LV                2  
Max PV                 0  
Cur PV                1  
Act PV                1  
VG Size                465.52 GiB  
PE Size                4.00 MiB  
Total PE               119173  
Alloc PE / Size        119161 / 465.47 GiB  
Free PE / Size          12 / 48.00 MiB  
VG UUID                8rdJMw-W2Dy-0BLL-tbSt-cfun-xq0c-hx2S2c  
  
tutonics@andromeda:~$
```

If we then run the command:

```
sudo vgextend ubuntu /dev/sdb1
```

You should see this message:

Volume group "ubuntu" successfully extended

Here's what the volume group looks like after that command:

```
tutonics@andromeda: ~  
tutonics@andromeda:~$ sudo vgdisplay ubuntu  
--- Volume group ---  
VG Name                ubuntu  
System ID  
Format                 lvm2  
Metadata Areas         2  
Metadata Sequence No   4  
VG Access               read/write  
VG Status               resizable  
MAX LV                 0  
Cur LV                 2  
Open LV                 2  
Max PV                 0  
Cur PV                 2  
Act PV                 2  
VG Size                 698.40 GiB  
PE Size                 4.00 MiB  
Total PE                178791  
Alloc PE / Size         119161 / 465.47 GiB  
Free PE / Size           59630 / 232.93 GiB  
VG UUID                 8rdJMw-W2Dy-0BLL-tbSt-cfun-xq0c-hx2S2c  
  
tutonics@andromeda:~$
```

As a side note, if you now do a `pvdisplay` on `/dev/sdb1` using:

```
sudo pvdisplay /dev/sdb1
```

You'll see something like this:

```
tutonics@andromeda:~$ sudo pvdisplay /dev/sdb1
--- Physical volume ---
PV Name                /dev/sdb1
VG Name                ubuntu
PV Size                232.88 GiB / not usable 2.18 MiB
Allocatable            yes
PE Size                4.00 MiB
Total PE               59618
Free PE                59618
Allocated PE           0
PV UUID                lX6Dz4-oV1L-21or-uElh-b90X-px3W-crK7LP
```

This tells you how many physical extents `/dev/sdb1` adds to the volume group `ubuntu`, and that all these physical extents are free and available for use in logical volumes (in the `ubuntu` group).

Extending A Logical Volume

If you have free space in your volume group, you can easily extend the root logical volume.

In our example, we shall extend the root filesystem "/" which is part of a default Ubuntu 12.10 install. The same principles apply when extending any logical volume.

After the logical volume has been extended, you must also extend the size of the underlying filesystem (which in our case is ext4). As you'll see shortly, we can do all of this as part of the same command.

To extend a logical volume, you can use either the **lvextend** command or the **lvresize** command. The **lvextend** command is considered a safer option because it won't allow you to reduce the size of the logical volume should you make a mistake when entering the command.

We can use the **--resizefs** option to automatically resize the underlying filesystem.

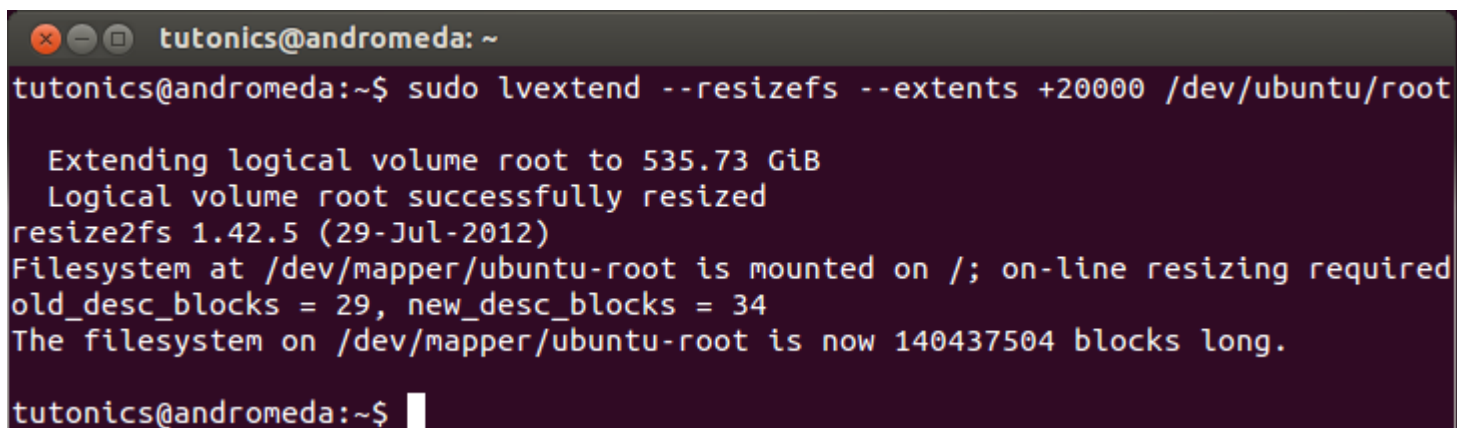
Purly for illustration purposes, we'll extend our 457.6GiB (491.34GB in hard disk terms) root logical volume (or another way of expressing the size is: 117146 logical extents).

Let's extend it by a further 20000 extents (which will extend it from 457.6GiB to 535.73GiB).

To do this and also resize the underlying ext4 filesystem, we'd use the command:

```
sudo lvextend --resizefs --extents +20000 /dev/ubuntu/root
```

The output of this command will look like this:

A terminal window titled 'tutonics@andromeda: ~' showing the execution of the command 'sudo lvextend --resizefs --extents +20000 /dev/ubuntu/root'. The output indicates that the logical volume 'root' is extended to 535.73 GiB, the filesystem is successfully resized, and the filesystem on '/dev/mapper/ubuntu-root' is now 140437504 blocks long.

```
tutonics@andromeda:~$ sudo lvextend --resizefs --extents +20000 /dev/ubuntu/root

  Extending logical volume root to 535.73 GiB
  Logical volume root successfully resized
resize2fs 1.42.5 (29-Jul-2012)
Filesystem at /dev/mapper/ubuntu-root is mounted on /; on-line resizing required
old_desc_blocks = 29, new_desc_blocks = 34
The filesystem on /dev/mapper/ubuntu-root is now 140437504 blocks long.

tutonics@andromeda:~$
```

If you'd rather extend by a specific size specified in Megabytes, Gigabytes, Terabytes, Petabytes, or Exabytes, you could use M,G,T,P, or E respectively (or the relevant letter in lower case). For example, the following command would extend by 20GiB (that is 20 Gigabytes in the sense of $20 \times 1024 \times 1024 \times 1024$ bytes, aka 20GiB).

```
sudo lvextend --resizefs --size +20G /dev/ubuntu/root
```

Furthermore, if you'd like to extend to a specific overall size, you would simply specify that size (no + used), e.g. to extend to 550GiB, use:

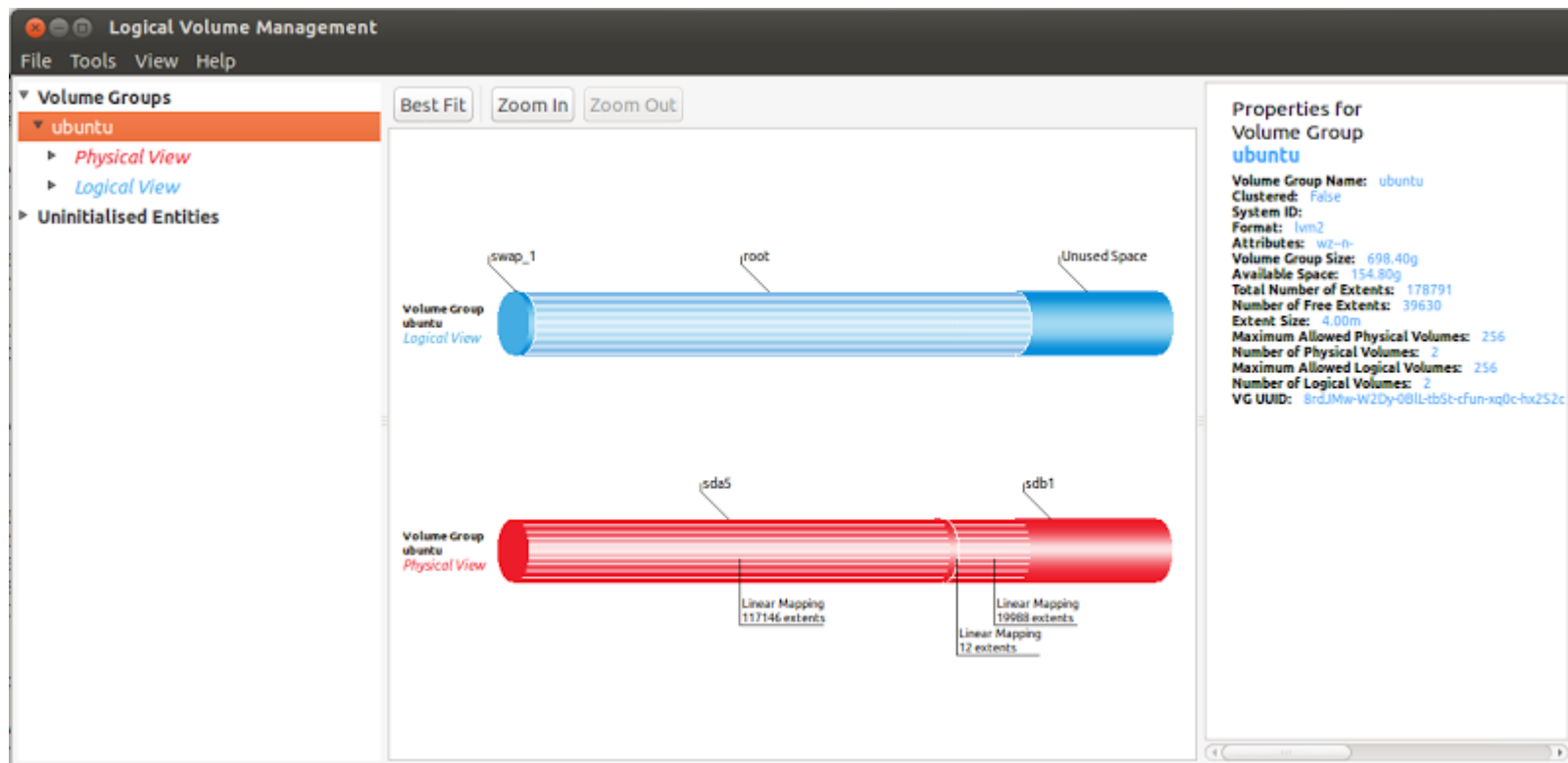
```
sudo lvextend --resizefs --size 550G /dev/ubuntu/root
```

Also note that if you wish to use physical extents from a particular physical volume to fulfil the request (i.e. you want to specify the underlying disk used) you can do so for any of the above commands by appending one or more physical volume names.

So for example, we had 12 free physical extents in our ubuntu volume group before we added the /dev/sdb1 physical volume to the group. When we extended the "root" logical volume by 20000 PE's, that 12 (residing on /dev/sda5) were used along with 19988 taken from /dev/sdb1. Had we wanted that all 20000 PE's from /dev/sdb1 we could have used:

```
sudo lvextend --resizefs --extents +20000 /dev/ubuntu/root /
```


For illustration purposes, the screenshot below shows how things look now for us (given that we didn't specify we wanted all 20000 physical extents from same physical volume).



If you're curious, take a look at the volumes under "Physical View" and Logical Views" in the system-config-lvm tool to see exactly how things look visually.

Creating A New Logical Volume

When you create a new logical volume, you create it in a volume group. That volume group must have sufficient free space (physical extents) to create your logical volume.

Note that when choosing the name of your logical volume, it is best not to use a "-" (i.e.dash/hyphen) in the name if you plan on using the logical volume with grub2. So for example, grub2 would have issues with finding my-lv but not with my_lv.

In our example here (which has nothing to do with grub2), we'll create a logical volume for our music called music_lv in the ubuntu volume group. We can see how much free space the volume group (called ubuntu) has by running the command:

```
sudo vgdisplay ubuntu
```

In our case, we see this in the output:

```
Free   PE / Size          39630 / 154.80 GiB
```

Lets create a logical volume called `music_lv` of size 30000 PE (physical extents).

In order to do this, we need to use the `lvcreate` command like so:

```
sudo lvcreate --extents 30000 -n music_lv ubuntu
```

As before, you can specify Megabytes, Gigabytes, Terabytes, Petabytes, or Exabytes using M, G, T, P, or E respectively along with the `--size` option. So we could have created a 100GiB volume like this (if we wanted that size):

```
sudo lvcreate --size 100G -n music_lv ubuntu
```

If the logical volume creation worked OK, you'll see the message:

```
Logical volume "music_lv" created
```

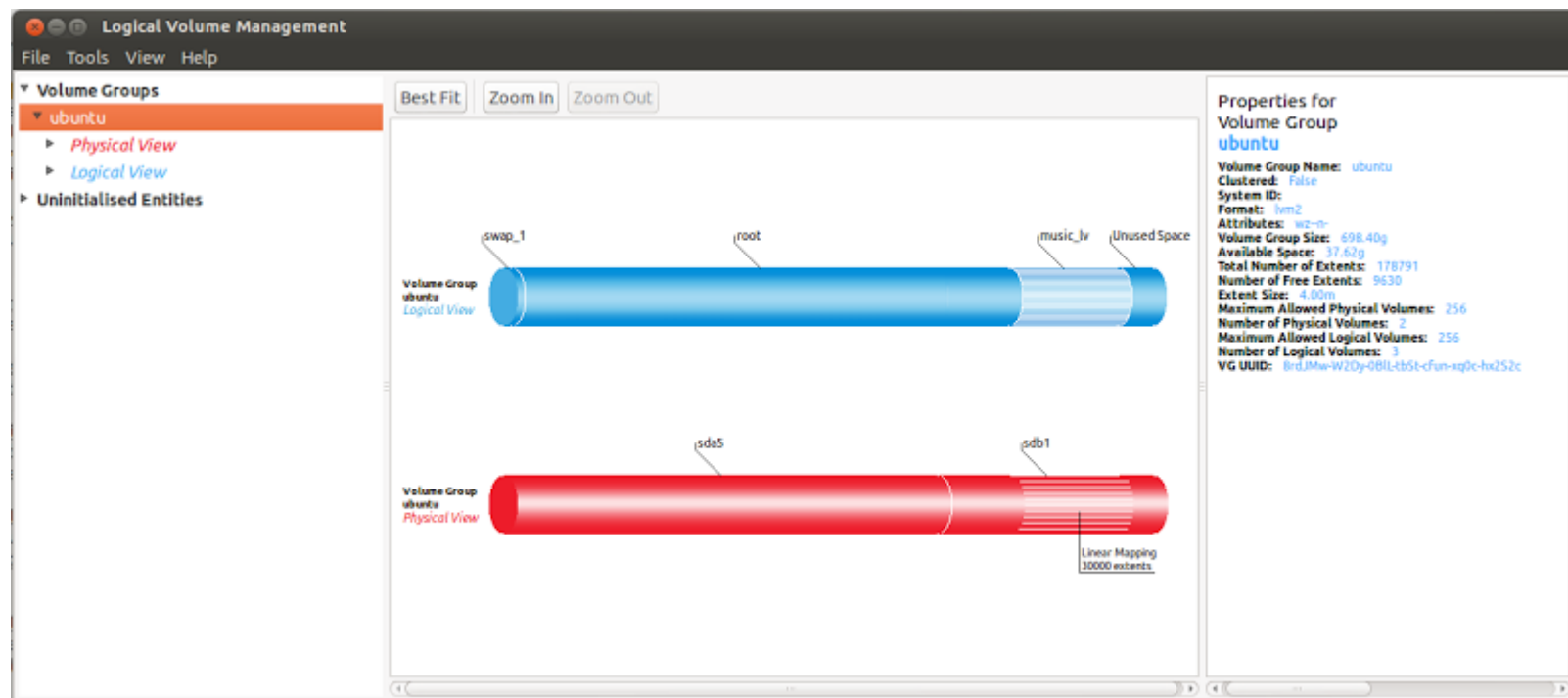
As things stand, we can't do much with `music_lv` yet because we have no filesystem on it. Lets create an ext4 filesystem on our new logical volume using the `mkfs.ext4` command:

```
sudo mkfs.ext4 /dev/ubuntu/music_lv
```

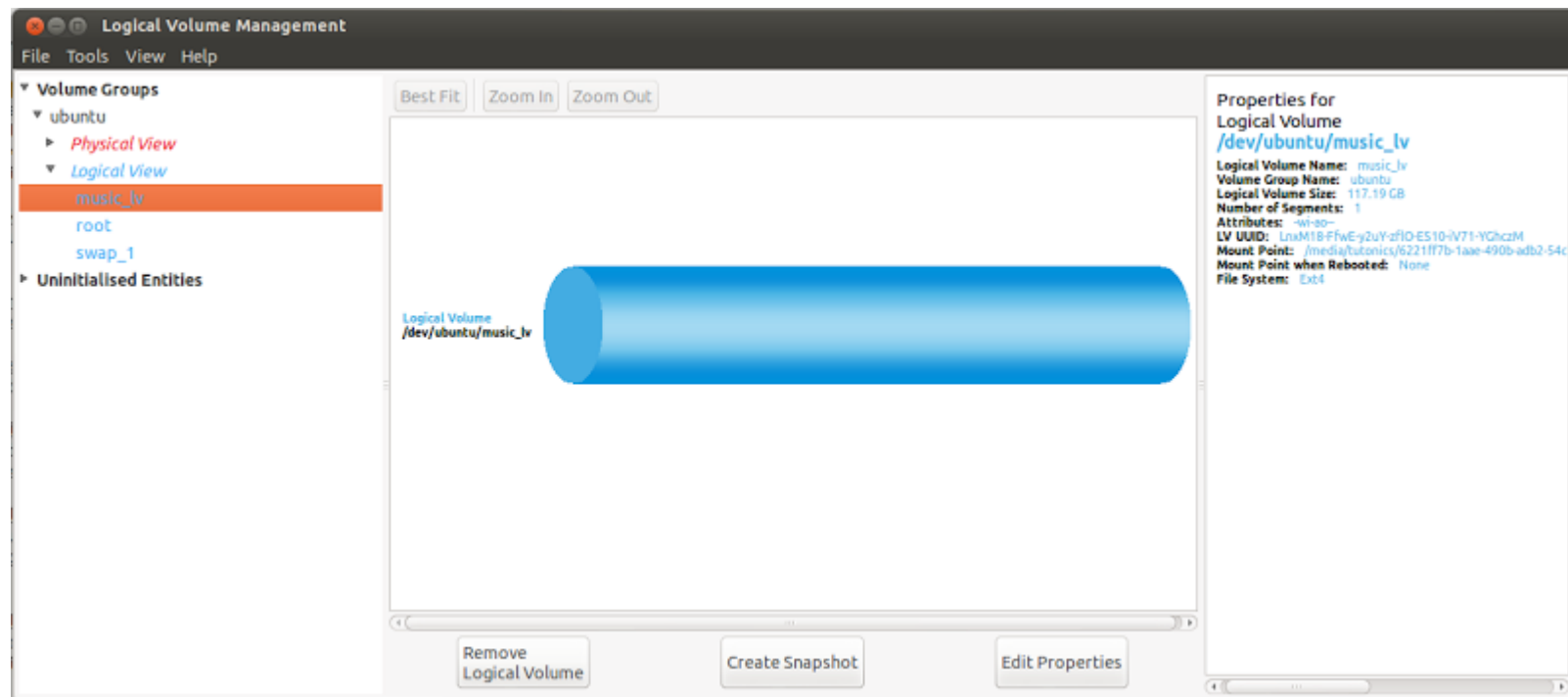
The output should look something like this:

```
tutonics@andromeda: ~  
tutonics@andromeda:~$ sudo mkfs.ext4 /dev/ubuntu/music_lv  
mke2fs 1.42.5 (29-Jul-2012)  
Filesystem label=  
OS type: Linux  
Block size=4096 (log=2)  
Fragment size=4096 (log=2)  
Stride=0 blocks, Stripe width=0 blocks  
7684096 inodes, 30720000 blocks  
1536000 blocks (5.00%) reserved for the super user  
First data block=0  
Maximum filesystem blocks=4294967296  
938 block groups  
32768 blocks per group, 32768 fragments per group  
8192 inodes per group  
Superblock backups stored on blocks:  
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,  
    4096000, 7962624, 11239424, 20480000, 23887872  
  
Allocating group tables: done  
Writing inode tables: done  
Creating journal (32768 blocks): done  
Writing superblocks and filesystem accounting information: done  
  
tutonics@andromeda:~$
```

And for illustration purposes, the screenshot below shows our new logical volume `music_lv` is part of the lvm layout:



A look at the "Logical View" for our new music_lv volume shows the following:



If you'd like Music in your home directory to use (mount to) music_lv logical volume, you could do the following (swap your user name for tutonics):

```
sudo mount /dev/ubuntu/music_lv /home/tutonics/Music/
```

Then sort out the permissions:

```
sudo chown tutonics /home/tutonics/Music
```

```
sudo chgrp tutonics /home/tutonics/Music
```

Notice in the above screenshot that the text to the right says "Mount point when rebooted: None". If you'd like this to be mounted when you boot your box, you should add an entry to `/etc/fstab`.

To get an idea as to what you could use as this `/etc/fstab` entry run:

```
cat /etc/mtab | grep music_lv
```

Shrinking/Reducing A Logical Volume

When you want to reduce the size of a logical volume, **you cannot be mounted on that volume**. Hence, if you wish to reduce the size of your "root" logical volume, you'll need to do it via a live disk.

In this example, we'll reduce the size of the `music_lv` that we just created above. Lets reduce it by 10000 PE's (Currently it consists of 30000, so after shrinking/reducing the logical volume, it will consist of 20000 physical

extents). Ensure the filesystem on the music_lv is not mounted. It had been mounted on /media/tutonics/6221ff7b-1aae-490b-adb2-54c7919d3286 so we can unmount it using the command:

```
sudo umount /media/tutonics/6221ff7b-1aae-490b-adb2-54c7919d
```

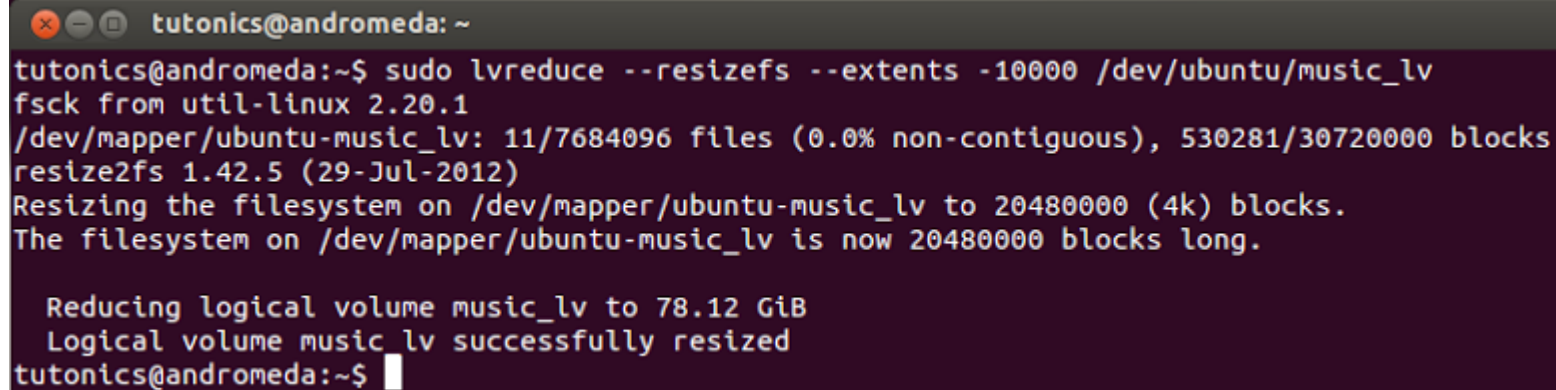
Now for a couple of golden rules to follow when reducing the size of a logical volume:

1. Always make sure you have removed enough data from the filesystem on that logical volume to allow for the size you want to reduce by, otherwise your data could be destroyed.
2. Always shrink the size of the filesystem before you shrink the logical volume, otherwise your data will be lost. (to be safe, we'll ask for both to be done as part of the same command).

OK, to carry out the reduction, use the `lvreduce` (or `lvresize`) command along with the `--resizefs` option as shown below:

```
sudo lvreduce --resizefs --extents -10000 /dev/ubuntu/music_
```

Here is a screenshot of the output from this command:



```
tutonics@andromeda: ~  
tutonics@andromeda:~$ sudo lvreduce --resizefs --extents -10000 /dev/ubuntu/music_lv  
fsck from util-linux 2.20.1  
/dev/mapper/ubuntu-music_lv: 11/7684096 files (0.0% non-contiguous), 530281/30720000 blocks  
resize2fs 1.42.5 (29-Jul-2012)  
Resizing the filesystem on /dev/mapper/ubuntu-music_lv to 20480000 (4k) blocks.  
The filesystem on /dev/mapper/ubuntu-music_lv is now 20480000 blocks long.  
  
Reducing logical volume music_lv to 78.12 GiB  
Logical volume music_lv successfully resized  
tutonics@andromeda:~$
```

As usual, if you want you can specify to reduce by a certain number of Gigabytes by using the `--size` option instead, e.g. to reduce by 20 GiB, use:

```
sudo lvreduce --resizefs --size -20G /dev/ubuntu/music_lv
```

or to reduce to a specific size, say 100GiB, use:

```
sudo lvreduce --resizefs --size 100G /dev/ubuntu/music_lv
```

In our case because we reduced by 10000 PE's (from 30K to 20K), our `music_lv` logical volume now looks like this:

```
tutonics@andromeda: ~  
tutonics@andromeda:~$ sudo lvsdisplay /dev/ubuntu/music_lv  
--- Logical volume ---  
LV Path                /dev/ubuntu/music_lv  
LV Name                 music_lv  
VG Name                 ubuntu  
LV UUID                 LnxM18-FfwE-y2uY-zf10-ES10-iV71-YGhczM  
LV Write Access         read/write  
LV Creation host, time  andromeda, 2012-11-27 20:27:03 +0000  
LV Status                available  
# open                  0  
LV Size                 78.12 GiB  
Current LE              20000  
Segments                1  
Allocation               inherit  
Read ahead sectors      auto  
- currently set to      256  
Block device             252:2  
  
tutonics@andromeda:~$
```

What's Next?

In the next part of our LVM guide, we'll show you how to create and use [snapshots](#).

Thanks to the programmers of LVM2 and also the authors of the [LVM-HOWTO](#)

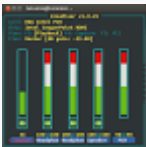
Thank you for reading this article. Please share if you liked it.



You might also like:



How to Change Update
Check Frequency &
Automatically Install
Fix No Sound / Turn Off
Security Updates
Auto Mute in Ubuntu



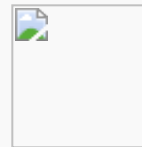
12.10



Ubuntu LVM Guide -
Part 2 (Snapshots)



Preparing VirtualBox
for Virtual OS
Installations



How to find out if your
Processor is 64-bit or
32-bit



Enable Ubuntu
Restricted Extras &

© 2017 Tutonics