Logical Volume Management

Before we go to far - if your needs are very simple, and you have enough space on your primary drive, then skip down to the SAMBA section, otherwise read on.

WARNING - everything in this section has the ability to completely and utterly destroy data - what you do with your data is your business - these steps, if not clearly understood, will result in unrecoverable data loss to the drives/devices/volumes being worked on - back up your drives before considering any of these steps

The Linux Logical Volume Manager (LVM) is a mechanism for virtualizing disks. It can create “virtual” disk partitions out of one or more physical hard drives, allowing you to grow, shrink, or move those partitions from drive to drive as your needs change. It also allows you to create larger partitions than you could achieve with a single drive.

Traditional uses of LVM have included databases and company file servers, but even home users may want large partitions for music or video collections, or for storing online backups. LVM and RAID 1 can also be convenient ways to gain redundancy without sacrificing flexibility.

This article looks first at a basic file server, then explains some variations on that theme, including adding redundancy with RAID 1 and some things to consider when using LVM for desktop machines.

LVM Basics

To use LVM, you must understand several elements.

First are the regular physical hard drives attached to the computer. The disk space on these devices is chopped up into partitions. Finally, a filesystem is written directly to a partition.

By comparison, in LVM, Volume Groups (VGs) are split up into logical volumes (LVs), where the filesystems ultimately reside

Each VG is made up of a pool of Physical Volumes (PVs). You can extend (or reduce) the size of a Volume Group by adding or removing as many PVs as you wish, provided there are enough PVs remaining to store the contents of all the allocated LVs. As long as there is available space in the VG, you can also grow and shrink the size of your LVs at will (although most filesystems don't like to shrink).

An Example LVM Stack

In the exercise below, we'll follow this design - it's here for teaching purposes only - in production this will work, but perhaps consider using xfs vs. ext4 for production, reasons there are in the advanced section

Section	Tool		
Mounts	mount	/var/share	/var/media
FileSystem	mkfs	ext4	xfs
Logical Volume	lv	/share	/media
Volume Group	vg	/dev/fileserver	
RAID (if applicable)	mdadm	/dev/md0	
Partitions	pv	/dev/sdb1	/dev/sdc1

Disks	fdisk	/dev/sdb	/dev/sdc

Basic LVM Setup

A simple, practical example of LVM use is a traditional file server, which provides centralized backup, storage space for media files, and shared file space for several family members' computers.

Flexibility is a key requirement; who knows what storage challenges next year's technology will bring?

For example, suppose your requirements now are:

20G - Large media file storage
20G - Shared files

Ultimately, these requirements may increase a great deal over the next year or two, but exactly how much and which partition will grow the most are still unknown.

Disk Hardware

Traditionally, a file server uses SCSI disks, but today SATA disks offer an attractive combination of speed and low cost. With USB3, it's even more so with external disks that are easy to attach (and easy to remove, which is not recommended if extending an LVM set to USB3)

SATA drives are not named like ATA drives (hda, hdb), but like SCSI (sda, sdb). Once the system has booted with SATA support, it has three physical devices mounted - the boot disk, which is generally, but may not be /dev/sda, and the two attached disks which are not part of the boot partition or file system.

For purposes of this discussion - the attached non-boot disks are;

/dev/sdb 32.0 GB
/dev/sdc 32.0 GB

Next, partition these for use with LVM. You can do this with fdisk by specifying the “Linux LVM” partition type 8e.

```
fdisk /dev/sdb

Command (m for help); <-- d to delete existing partition

Command (m for help): <-- n for new partition

Command action
   e   extended
   p   primary partition (1-4) <-- p for primary
Partition number (1-4): <-- 1
First cylinder (1-10443, default 1): <-- <ENTER> to accept defaults
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-10443, default 10443): <-- <ENTER> to accept defaults

Command (m for help): <-- t for type
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): <-- w to write the partion back to disk
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

Now we do the same for the other hard disk /dev/sdc:

The finished product looks like this for my test drives:

```
sudo fdisk -l /dev/sdc
Disk /dev/sdc: 29.8 GiB, 32008830976 bytes, 62517248 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
Disklabel type: dos
Disk identifier: 0x5c6842c1

Device     Boot Start      End  Sectors  Size Id Type
/dev/sdc1        2048 62517247 62515200 29.8G fd 8e  Linux LVM
```

Note that the partition type here is '8e', or “Linux LVM.”

Creating a Virtual Volume

We have 2 drives with Linux LVM Partition - sdb1 and sdc1

Create the LVM PV Group

Initialize each of the disks using the **pvcreate** command:

```
pvcreate /dev/sdb1 /dev/sdc1
```

This sets up all the partitions on these drives for use under LVM, allowing creation of volume groups. To examine available PVs, use the **pvdisplay** command.

Create the LVM Volume Group

This system will use a single-volume group named fileserver:

```
vgcreate fileserver /dev/sdb1 /dev/sdc1
```

Use **vgdisplay** to see the newly created “fileserver” VG with the two drives stitched together.

Create the LVM Logical Volumes

Now create the logical volumes within them:

```
lvcreate --name media --size 20G fileserver
lvcreate --name share --size 20G fileserver
```

Use **lvdisplay** to see the newly created logical volumevg

Without LVM, you might allocate all available disk space to the partitions you're creating, but with LVM, it is worthwhile to be conservative, allocating only part the available space to the current requirements.

As a general rule, it's easier to grow a filesystem than to shrink it, so it's a good strategy to allocate exactly what you need today, and leave the remaining space unallocated until your needs become clearer.

This method also gives you the option of creating new volumes when new needs arise (such as a separate encrypted file share for sensitive data).

Now you have several nicely named logical volumes at your disposal:

/dev/fileserver/media
/dev/fileserver/share

Summarizing where we are to this point - again, we have Logical Volumes over a Volume Group sitting on a Physical Volume Group spread over two Physical Drive Partitions - see below;

The Logical Volumes

```
test@testbox:~$ sudo lvdisplay
  --- Logical volume ---
  LV Path                /dev/fileserver/share
  LV Name                share
  VG Name                fileserver
  LV UUID                xNEcAi-Y4s6-mt9B-6ic6-iIOT-hN4i-gIkNg6
  LV Write Access        read/write
  LV Creation host, time testbox, 2016-05-20 22:25:42 -0700
  LV Status              available
  # open                 0
  LV Size                20.00 GiB
  Current LE             5120
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           252:0

  --- Logical volume ---
  LV Path                /dev/fileserver/media
  LV Name                media
  VG Name                fileserver
  LV UUID                enRNSF-8awM-QQ73-q4mV-7vDu-s6mN-Vj9E3H
  LV Write Access        read/write
  LV Creation host, time testbox, 2016-05-20 22:26:08 -0700
  LV Status              available
  # open                 0
  LV Size                20.00 GiB
  Current LE             5120
  Segments               2
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           252:1
```

The Volume Group

```
test@testbox:~$ sudo vgdisplay
  --- Volume group ---
  VG Name               fileserver
  System ID
  Format              lvm2
  Metadata Areas       2
  Metadata Sequence No  5
  VG Access            read/write
  VG Status            resizable
  MAX LV               0
  Cur LV               2
  Open LV              0
  Max PV               0
  Cur PV               2
  Act PV               2
  VG Size              59.62 GiB
  PE Size              4.00 MiB
  Total PE             15262
  Alloc PE / Size      10240 / 40.00 GiB
  Free  PE / Size      5022 / 19.62 GiB
  VG UUID              f5Z2w1-PSZi-9K3G-9RTp-1jNx-gj0q-WugTqD
```

And the Physical Volume Group

```
test@testbox:~$ sudo pvdisplay
  --- Physical volume ---
  PV Name               /dev/sdb1
  VG Name               fileserver
  PV Size               29.81 GiB / not usable 0
  Allocatable           yes (but full)
  PE Size               4.00 MiB
  Total PE              7631
  Free PE               0
  Allocated PE          7631
  PV UUID               zaNOx1-fS0k-wNQ1-DGpq-yfNg-kc3e-j5toa1

  --- Physical volume ---
  PV Name               /dev/sdc1
  VG Name               fileserver
  PV Size               29.81 GiB / not usable 0
  Allocatable           yes
  PE Size               4.00 MiB
  Total PE              7631
  Free PE               5022
  Allocated PE          2609
  PV UUID               slUCwq-Gw81-FcbB-WF06-tt33-EQ27-9rT737
```

Note - here's a hint that we've been quietly building quotas for the shares - not on a user basis, but on a share basis - and this keep linux from running off the end of the pier if it runs out of disk space - it'll panic a bit, but you won't crash the entire volume set, it'll go read-only if the limits are hit, which is a good indication to see what's going on - another reason why LVM is a good thing)

Selecting Filesystems

Now that the logical volumes are created, the next step is to put filesystems on them. However, there are many types of filesystems. How do you choose?

- For typical desktop filesystems, you're probably familiar with ext4.
- ext4's balance of performance, robustness, and recovery speed makes it a fine choice for general purpose use. Because ext4 has been the default for such a long time, ext4 is also a good choice if you want great reliability.
- For storing backups, reliability is much more important than speed.
- The major downside to ext4 is that to grow (or shrink) the filesystem, you must first unmount it.
- However, other filesystems provide advantages in certain situations, such as large file sizes, large quantities of files, or on-the-fly filesystem growth.
- Because LVM's primary use is for scenarios where you need extreme numbers of files, extremely large files, and/or the need to resize your filesystems, the following filesystems are well worth considering.
- For large numbers of small files, XFS is an excellent choice, and it was designed specifically with LVM in mind.
- If your system is recording or processing video at the same time, delays may cause dropped frames or other glitches. XFS is better choice in this situation
- XFS has the edge due to greater reliability and better general performance.

Since we're building this document around Ubuntu, check their site for more information on the various file systems available for use - ext4, xfs are native to Ubuntu 16.04 and well proven

With all these considerations in mind, the intent here is to keep things as simple as possible for the moment - again, this is only as an example for the walk-thru

Format the partitions on the LV's as follows:

```
mkfs.ext4 /dev/fileserver/share
mkfs.xfs /dev/fileserver/media
```

NOTE - this is just an example - I typically use xfs on all my LVM volumes - in the advanced section, all logical volumes are xfs, and before you start, review that section, and I think you'll agree

Mounting the Logical Volumes

Now to create mount points in the primary file system

```
sudo mkdir /var/share
sudo mkdir /var/media
```

Now we can test mount the logical volumes

```
sudo mount /dev/fileserver/share /var/share
sudo mount /dev/fileserver/media /var/media
```

you can check by doing a df -h and see that the file system has the new volumes and sizes

```
/dev/mapper/fileserver-share   20G   44M   19G   1% /var/share
/dev/mapper/fileserver-media   20G   33M   20G   1% /var/media
```

Finally, to persistently mount the file systems, first add the following lines to /etc/fstab:

```
/dev/fileserver/share    /var/share     ext4   rw,noatime    0 0
/dev/fileserver/media    /var/media      xfs   rw,noatime    0 0
```

And reboot the system

Adding Reliability With RAID

So far, this LVM example has been reasonably straightforward.

That's right - it's a spanned drive, which is even more worrisome that a RAID0, as the read/write performance is limited to the speed of the drive within the span that data is being read from/written to, and there is no indication of any issues with the drive.

So let's convert our spanned drive to a RAID array

First, any data that is in those logical volumes - back it up, as this is a destructive process;

Second, if these are shared, stop any processes that might be using them

Then unmount the volumes

```
umount /dev/fileserver/media
umount /dev/fileserver/share
```

Check with df to make sure they're unmounted

Now we disassemble the sets... going it bit faster this time

```
lvremove /dev/fileserver/media
lvremove /dev/fileserver/share
vgremove fileserver
pvremove /dev/sdb1 /dev/sdc1
```

And we're back to two devices on the bus - by removing the LV's, we also remove the filesystem that overlaid them

Update the /etc/fstab and comment out the two LVM shares - we will be using the same two mounts after we build the array

Setting up a Software RAID array

Then, change the partition type on these two drives, using filesystem type fd (Linux raid autodetect) -

Delete the old partitions if needed, and create New Primary partitions - below we're setting up a RAID0, but you can change --level switch to 1,5,6,10 depending on number of disks, and you're intents.. this is only meant as an example.

Write the changes, and do an fdisk -l to check

```
mdadm --create --verbose /dev/md0 --level=0 --raid-devices=2 /dev/sdb1 /dev/sdc1
```

Update the mdadm.conf file

```
mdadm --detail --scan >> /etc/mdadm/mdadm.conf
```

Fix Me! - http://www.ducea.com/2009/03/08/mdadm-cheat-sheet/

And add it back to the LVM

Moving quickly

```
pvcreate /dev/md0
vgcreate fileserver /dev/md0
lvcreate --name share --size 20G fileserver
lvcreate --name media --size 20G fileserver
```

and you can verify that we're back to where we were...

```
lvdisplay
```

Create the filesystem on the logical volumes

```
mkfs.xfs /dev/fileserver/media
mkfs.ext4 /dev/fileserver/media
```

Redo the mounts

```
mount /dev/fileserver/media /var/media
mount /dev/fileserver/share /var/share
```

check df to see that they're mounted - if good, go back and uncomment the lines for /var/share and /var/media in the /etc/fstab

Reboot, and you're set

LVM advanced examples

LVM, like many other enterprise class technologies, brings new flexibility to the linux desktop.

While the previous walk-thru is small in scope, the real power of LVM is dealing with a large volume group, and being able to manage the Physical Volumes, Logical Volumes, and maintain your data.

LVM Growth

Suppose that over the next year, the storage system fills up and needs to be expanded. Initially, you can begin allocating the unallocated space.

In our previous example, we had a 59GB Volume Group (VG) and only 20GB allocated to /var/media - we can grow this out..

For instance, to increase the amount of space available for media files from 20GB to 25GB, run a command such as:

```
umount /dev/fileserver/media
lvextend -L25G /dev/fileserver/media
mount /dev/fileserver/media /var/media
xfs_growfs /dev/fileserver/media
```

Note that we're not increasing the size of /var/media, but the actual LVM logical volume that it's mounted from the Operating System, so the OS sees it, and uses it.

Note - there is a -D switch on xfs_growfs, setting the limit in blocks (not bytes), if it is not set, then xfs_growfs will expand the filesystem to the limit of the volume.

WARNING - one should <u>never</u> reduce the size of an LVM Logical Volume, and xfs, while it can extend a partition/volume, it cannot shrink - to reduce the size of any partition on any filesystem, it's always a backup, and restore to the new volume size

Growth via MDADM - RAID

Leveling up - this is pretty complicated, but a structured approach works - takes time, and of course, always have a backup of your data...

WARNING - we're now mucking about deep inside the LVM stack - things might go wrong, may go wrong - so always, I repeat, always have a backup before doing operations like this

Let's say we have an LVM Logical Volume Group running on top of a RAID

As an example - we have 4 1TB drives running, and we want to swap them out for 4 4TB drives - we can do this. But it will take a bit of time to migrate and grow, as we do one disk at a time, and let MDADM sync the drives before going to the next disk in the arrary

Note - as a RAID5, 4*1TB = 3TB as we lose the 1TB for parity

Here's the lineup for the example - /dev/md0 is a member of the fileserver volume group (VG)

The RAID5 device is /dev/md0

The members of /dev/md0

```
/dev/sdb1
/dev/sdc1
/dev/sdd1
/dev/sde1
```

We first unmount the logical volumes across the entire LVM set.

```
umount /dev/fileserver/share
umount /dev/fileserver/media
```

With RAID5, the migration is straightforward, as we use RAID5's ability to run in degraded mode if we have a failed drive

Use mdadm to mark one single drive of the RAID5 mirrors as failed, and then remove it:

```
mdadm --manage /dev/md0 --fail /dev/sda1
mdadm --manage /dev/md0 --remove /dev/sda1
```

Pull out the sda hard drive and replace it with the new 4TB drive.

Go into fdisk, and partition the drive (see above) making sure to split the physical drive into two partitions, sda1 as the same size as the old 1TB sda1, and add new sda2 partion, which we add the remaining balance of sda to sda2 (approx 2.8TB)

Now we manage sda1 back to /dev/md0 - remember, that we marked sda1 as failed

```
mdadm --manage /dev/md0 --add /dev/sda1
```

MDADM will then add /sda1 back to the array and start rebuilding the RAID5 set - once it's done, do the same with /sdb1 and so forth...

In the end of all this, we have 8 sets of partitions the sd<d,c,d,e>1 set, and the remainders - so we take the second set of partitions, and build that into a new /dev/md1 device

```
mdadm --create --verbose /dev/md1 --level=5 --raid-devices=4 /dev/sdb2 /dev/sdc2 /dev/sdd2
/dev/sde2
```

So now we have a second MD RAID5 set - and this one is 12TB (remember that md0 is already 4TB recreated from the previous 4 1TB drives)

Let md1 finish it's sync up - can check mdstat to see the progress - when it's done...

we add them to the Physical Volume group (pv)

```
pvcreate /dev/md1
```

and extend the volume group

```
vgextend fileserver /dev/md1
```

Now the fileserver Volume Group has 12TB of storage compared to the previous 3TB of storage, and we can use LV to grow the file systems...

The previous allocations were 750GB for /share and 2TB for media, and now we want to put 3TB for /share, and 7TB /media (remember, we don't allocate the entire VG space to keep a bit in reserve)

```
lvextend -L7T /dev/fileserver/media
lvextend -L3T /dev/fileserver/share
```

And now we have 10TB of space - and we can grow the file systems according - and this is why xfs is much better than ext4

Mount the file systems again - as they were unmounted before we started swapping out disks...

```
mount /dev/fileserver/media /var/media
mount /dev/fileserver/share /var/share
```

Now we tell xfs to grow... but the filesystem must be mounted before we do this...

```
xfs_growfs /dev/fileserver/media
xfx_growfs /dev/fileserver/share
```

And xfs will grow the filesystem to the extent of the volume.

What I'm getting at is that with LVM, we can definitely swizzle things around with Physical Volume, and Logical Volume all within the same Volume Group set - it's a very powerful solution, and one that enterprises use all the time.

Always do your homework, have a plan, and do some research - this is Linux after all, and changes happen and API's evolve - and always have a backup plan.