Teoría File System

Logical Volume Manager (LVM) versus standard partitioning in Linux

https://www.redhat.com/sysadmin/lvm-vs-partitioning

Use this guide to integrate the flexibility, scalability, and increased features of LVM into your server storage strategies. Traditional partitioning is good, but LVM is better.

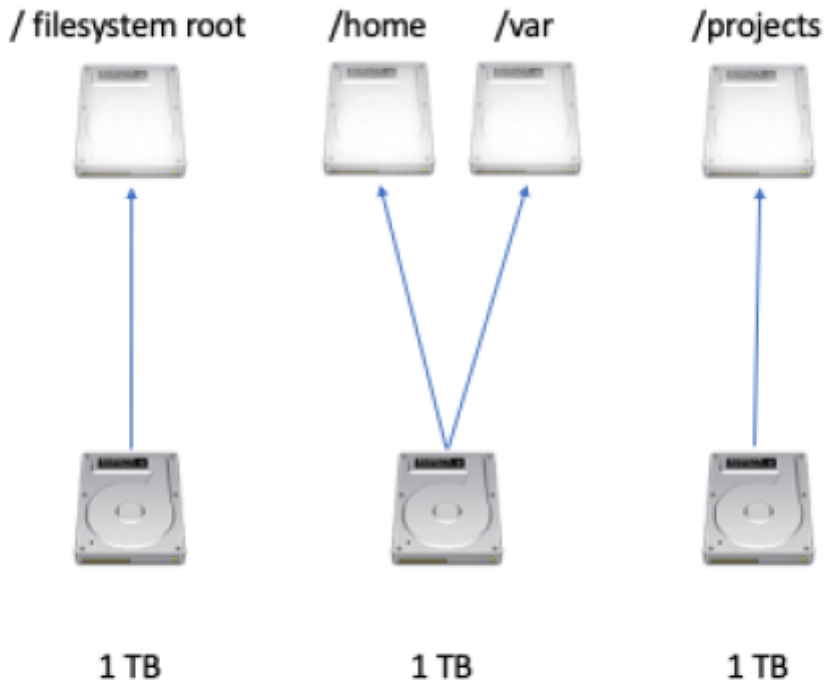
Server storage capacity has been managed via disk drive sizes and partition configurations for decades. Clearly, those strategies work well and are reliable. However, there are many benefits to rethinking storage management on local servers. This article compares standard storage management and partitioning to Logical Volume Manager (LVM). It also demonstrates some basic commands for each approach.

Traditional storage management

I use the phrase traditional storage management to describe the process of partitioning, formatting, and mounting storage capacity from a basic hard disk drive. I contrast this standard partitioning with an alternative method called Logical Volume Manager, or LVM.

Storage space is typically managed based on the maximum capacity of individual hard disk drives. The result is that when a sysadmin thinks about storage, they do so based on each drive. For example, if a server has three hard disk drives of 1 TB each, the sysadmin considers the storage literally, I have three 1 TB drives to work with.

An administrator thinks of standard partitions based on individual drive capacity.



Three 1 TB hard drives with partitions and mount points. The partitions are entirely contained on the individual hard disk drives.

Let's very quickly review traditional storage management. Here is a sample scenario:

1. Install a new hard disk drive

Purchase a one terabyte (1 TB) hard disk drive, and then physically install it into the server.

2. Partition the drive

Use fdisk or gparted to create one or more partitions. It's important to note that the partitions cannot consume more than the total 1 TB of disk capacity.

Example fdisk command:

fdisk /dev/sdb
I won't cover the syntax for fdisk in this article, but assume I created a single partition that consumes the entire 1 TB disk. The partition is /dev/sdb1.

Display the capacity by using the /proc/partitions and lsblk content:

cat /proc/partitions
lsblk

3. Create a filesystem

Create a filesystem on the new partition by using the mkfs command. You could use ext4 or RHEL's default XFS filesystem.

```
# mkfs.ext4 /dev/sdb1
```

While XFS is Red Hat's default, it may not be as flexible when combined with LVM as ext4. XFS filesystems can easily be extended but not reduced. I'll expand on that idea further toward the end of the article.

4. Create a mount point

The rest of this process is relatively standard. First, create a directory to serve as a mount point. Next, manually mount the partition to the mount point.

```
# mkdir /newstorage
# mount /dev/sdb1 /newstorage
```

5. Confirm the storage capacity

Use the du command to confirm the storage space is accessible and of the expected size.

```
# du -h /newstorage
```

Note: The -h option displays the output of du in capacity terms that are easy for humans to understand, such as GB or TB.

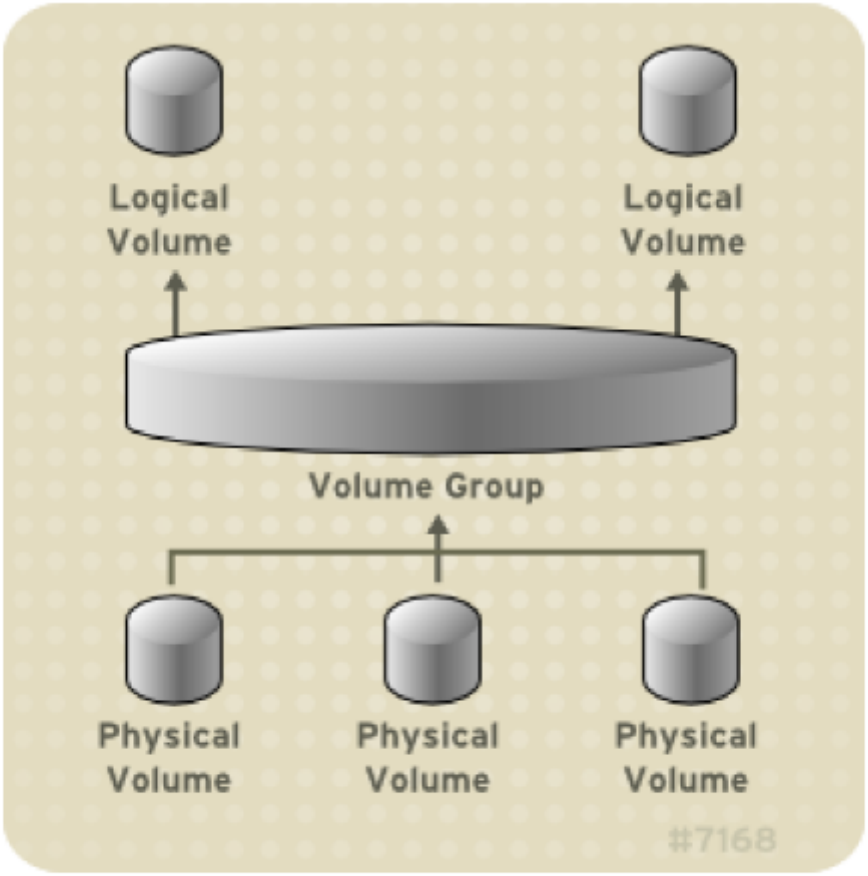
6. Configure the space to mount at boot

Edit the /etc/fstab file to mount the filesystem at boot. If you need a reminder on /etc/fstab, check out Tyler Carrigan's article An introduction to the Linux /etc/fstab file here on Enable Sysadmin.

Logical Volume Manager (LVM)

Traditional storage capacity is based on individual disk capacity. LVM uses a different concept. Storage space is managed by combining or pooling the capacity of the available drives. With traditional storage, three 1 TB disks are handled individually. With LVM, those same three disks are considered to be 3 TB of aggregated storage capacity. This is accomplished by designating the storage disks as Physical Volumes (PV), or storage capacity useable by LVM. The PVs are then added to one or more Volume Groups (VGs). The VGs are carved into one or more Logical Volumes (LVs), which then are treated as traditional partitions.

An administrator thinks of LVM as total combined storage space.



Three hard disk drives are combined into one volume group that is then carved into two logical volumes.

1. Install a new hard disk drive

Obviously, there needs to be a storage disk available. Just as we saw above, you must physically install a drive in the server.

2. Designate Physical Volumes

Physical Volumes (PV) are disks or partitions that are available to LVM as potential storage capacity. They have identifiers and metadata that describes each PV. It is interesting to note that, as opposed to RAID, PVs do not have to be the same size or or on disks that are the same speed. You can mix and match drive types to create PVs. To implement LVM, first designate a drive as a Physical Volume.

Command to create a PV:

pvcreate /dev/sdb1
pvcreate /dev/sdc
These two command examples are slightly different. The first command designates partition 1 on storage disk b as a PV. The second command sets the total capacity of storage disk c as a PV.

Display PV capacity and additional information:

pvdisplay
This command displays all of the Physical Volumes configured on the server.

3. Manage Volume Groups

Once one or more of the disks are available to LVM as Physical Volumes, the storage capacity is combined into Volume Groups (VGs). There may be more than one VG on a server, and disks may be members of more than one VG (but PVs themselves may only be members of one VG).

Use the vgcreate command to create a new Volume Group. The VG must have at least one member. The command syntax is:

vgcreate name-of-new-VG PV-members
Use the following command to create a Volume Group named vg00 with /dev/sdb1 and /dev/sdc as members:

vgcreate vg00 /dev/sdb1 /dev/sdc
Display information for a VG named vg00:

vgdisplay vg00

4. Manage Logical Volumes

The VG can be subdivided into one or more Logical Volumes (LVs). These Logical Volumes are then used as if they were traditional partitions. The VG has a total capacity, and then some part of that capacity is allocated to a Logical Volume.

The lvcreate command carves storage capacity from a VG. There are a few options to be aware of.

Option	Description
-n	Name of LV - ex. sales-lv
-L	Size in G or T - ex. 10G
-q	Quiet, suppresses command output
-v	Verbose mode providing additional details

lvcreate options
Option Description
-n Name of LV - ex. sales-lv
-L Size in G or T - ex. 10G
-q Quiet, suppresses command output
-v Verbose mode providing additional details
The syntax for the lvcreate command is as follows:

lvcreate -L size -n lvname vgname
Here is the command to create a 10 GB Logical Volume named sales-lv carved from the vg00 Volume Group:

lvcreate -L 10G -n sales-lv vg00
As you recall, we created the vg00 Volume Group from two Physical Volumes, /dev/sdb1 and /dev/sdc. So, in summary, we combined the capacity of /dev/sdb1 and /dev/sdc into vg00, then carved a Logical Volume named sales-lv from that aggregated storage space.

You can use the lvdisplay command to see the Logical Volume's configuration.

lvdisplay /dev/vg00/sales-lv

5. Apply a filesystem and set a mount point

Once the LV is created, it is managed as any other partition. It needs a filesystem and a mount point, just like we configured in the standard partition management section above.

Run the mkfs.ex4 command on the LV.
Create a mount point by using mkdir.
Manually mount the volume using the mount command, or edit the /etc/fstab file to mount the volume automatically when the system boots.
Use the df -h command to verify the storage capacity is available.
[You might also like: Introduction to Logical Volume Manager]

Scaling capacity

At this stage, we've seen the configuration of LVM, but we really have not yet been able to see the many benefits. One of the benefits of LVM configurations is the ability to scale storage capacity easily and quickly. Usually, of course, sysadmins need to scale up (increase capacity). It's worthwhile to note that you can also scale storage capacity down with LVM. That means that if storage capacity is over-allocated (you configured far more storage than you needed to), you can shrink it. I will cover both scenarios in this section.

Let's start with increasing capacity.

Increase capacity

You can add storage capacity to the Logical Volume. This is useful if the users consume more space than you anticipated. The process is pretty logical:

Add a disk and configure it as a PV.
Add it to a VG.
Add the capacity to the LV and then extend the filesystem.

1. Install a storage disk and then configure it as a PV

o increase capacity, install a new disk and configure it as a PV, as per the steps above. If there is already a disk with free space available, you can certainly use that, as well.

Here is a reminder of the command to create a PV:

pvcreate /dev/sdb2
In this case, I am designating partition 2 on disk /dev/sdb as the new PV.

2. Add space to the VG

Once the new capacity is designated for LVM, you can add it to the VG, increasing the pool's size.

Run this command to add a new PV to an existing VG:

vgextend vg00 /dev/sdb2
Now the storage pool is larger. The next step is to add the increased capacity to the specific Logical Volume. You can allocate any or all of the PV storage space you just added to the pool to the existing LV.

3. Add space to the LV

Next, add some or all of the new VG storage space to the LV that needs to be expanded.

Run the lvextend command to extend the LV to a given size:

lvextend -L3T /dev/vg00/sales-lv
Run the lvextend command to add 1 GB of space to the existing size:

lvextend -L+1G /dev/vg00/sales-lv

4. Extend the file system to make the storage capacity available

Finally, extend the file system. Both ext4 and XFS support this ability, so either filesystem is fine.

Unmount the filesystem by using the umount command:

umount /newstorage
Here is the basic command for ext4:

resize2fs /dev/vg00/sales-lv 3T

Reduce capacity

Reducing storage space is a less common task, but it's worth noting. The process occurs in the opposite order from expanding storage.

Note: XFS filesystems are not actually shrunk. Instead, back up the content and then restore it to a newly-resized LV. You can use the xfsdump utility to accomplish this. The ext4 filesystem can be reduced. That's the filesystem I focus on in this section. As we saw above with extending the filesystem, the volume must be unmounted. The exact command will vary depending on your LV name.

umount /dev/vg00/sales-lv

1. Shrink the filesystem

Next, use the resize2fs command to reduce the filesystem size. It is recommended that you run fsck on ext4 filesystems before shrinking them. It is also recommended that you back up the data on the LV in case something unexpected occurs.

Here is an example of shrinking an ext4 filesystem:

resize2fs /dev/vg00/sales-lv 3T
Note: You cannot shrink a filesystem to a size smaller than the amount of data stored on it.

2. Reduce the LV

Use the lvreduce command to shrink the storage space allocated to the LV. This returns the potential storage capacity to the VG.

lvreduce -L 2T vg00/sales-lv
It is critical to realize that the above command sets the sales-lv at 2T. It does not remove two terabytes from the existing LV. It configures the LV at two terabytes. It is possible to tell lvreduce to subtract an amount of space from the existing capacity by using a very similar command:

lvreduce -L -2T vg00/sales-lv
In this case, I added a - (dash) before the 2T size, indicating I want that quantity of space subtracted from the existing sales-lv capacity. The difference between the two commands is small but important.

You now have the returned capacity to the VG for use in another LV. You can use the extend commands discussed earlier to reallocate this capacity. The VG can also be shrunk.

Flexibility

Capacity can also be easily reallocated with LVM. You can reduce capacity in one VG and add it to another. This is accomplished by shrinking the filesystem and then removing the LV from the VG. Let's say you have a server with 10 TB of capacity. Using the above processes, you have created two LVs of 5 TB each. After a few weeks, you discover that you should have created LVs of 7 TB and 3 TB instead. You can remove 2 TB of capacity from one of the Volume Groups and then add that capacity to the other VG. This is far more flexibility than traditional partitioning offers.

LVM also supports RAID configurations, mirroring, and other advanced settings that make it an even more attractive solution. Tyler Carrigan's article Creating Logical Volumes in Linux with LVM has some good information on striping and mirroring Logical Volumes.

Wrap up

Logical Volume Manager is a great way to add flexibility to your storage needs, and it's really not much more complicated than traditional drive management. There is great documentation at Red Hat's site, and the official training courses cover it very effectively, as well.

One of the things that I appreciate about LVM is the logical command structure. The majority of the management commands are related and therefore relatively easy to remember. The point of the following table is not to summarize the commands, but rather for you to notice that the commands are all very similar and therefore user-friendly:

Similar commands are used to manage LVM components

Command	**Description**
pvcreate	Create physical volume
pvdisplay	Display physical volume information
pvs	Display physical volume information
pvremove	Remove physical volume
vgcreate	Create volume group
vgdisplay	Display volume group information
vgs	Display volume group information
vgremove	Remove volume group
vgextend/vgreduce	Extend or reduce volume group

lvcreate	Create logical volume
lvdisplay	Display logical volume information
lvs	Display logical volume information
lvremove	Remove logical volume
lvextend/lvextend	Extend or reduce logical volume

The next time you are standing up a local file server, consider using LVM instead of traditional storage management techniques. You may thank yourself months or years down the line as you need to adjust the server's storage capacity.

Revision #2
Created 13 December 2022 14:12:33 by Federico Alonso
Updated 13 December 2022 14:23:23 by Federico Alonso