8.3.1 Logical Volume Manager (LVM)

Click one of the buttons to take you to that part of the video.

Logical Volume Manager (LVM) 0:00-0:48

In this lesson, we're going to discuss something that's really cool. It's called logical volume management, which most folks just call LVM. LVM provides an alternative that you can use when you're partitioning the hard disk drives in your Linux system.

Basically what it does is create an alternative to the traditional process of creating disk partitions and mounting them in Linux file system. Instead of defining and creating partitions, what you do is define volume groups from all the various storage devices in your system.

Then, from the volume group, we allocate space to specific logical volumes, which are managed by the logical volume manager. Then, instead of mounting a partition in the file system, what you do is mount logical volumes at the various mount points in your file system.

LVM Flexibility 0:47-3:47

The great thing about LVM is the fact that it provides you with a huge degree of flexibility when you're allocating space on your system. Let's take a look at an example here.

On this Linux system, I have two LVM volumes. The first one is mounted in /opt, where various applications are installed; and the other one is mounted at /var, where I store my email queue, or I store my log files, where I store my print queues, and so on.

Because of the type of data that's stored in /var, it's very possible that the logical volume that I have mounted in /var is going to start running out of disk space at some point. Maybe I have a ton of users on this system. They send a lot of email, they send a lot of print jobs, and my log files are growing, so the size required for this volume is going to increase over time.

On the other hand, my /opt directory really doesn't need a whole lot of space. Once I've got my applications installed, I need only a little bit of space for a few minor changes. I don't need a ton of space here.

What we can do using LVM is we can actually take some of the space that was allocated to the logical volume mounted at /opt and move it to the logical volume mounted at /var. The whole process can be done seamlessly, transparently while the system remains running and servicing users.

Could you do this with traditional disk partitions? Yes, but would it be difficult? Yes, it would be a pain in the neck. Because basically, what I would have to do to reallocate space from one partition to another is backup both partitions and then cross my fingers, say a prayer, and resize the partitions, and then restore the data from backup. Hopefully none of the data would get mangled during the process.

In addition, LVM also allows you to dynamically add space to the system. For example, suppose I have a logical volume and I have mounted it at the /home directory. This is where all of my end users' /home directories reside. Let's suppose that on my Linux system, I have a logical volume defined with LVM and I've mounted it in the /home directory.

That means all of my end users' /home directories are stored inside of this logical volume. Let's suppose that we are now starting to run out of space. This is a very common occurrence as users download and create very large files.

To add capacity to this logical volume, all I have to do is install a new hard disk drive in the system and then allocate all of the space on that hard drive to the logical volume mounted at the /home. When I do, the size of that volume is instantly increased.

I didn't have to back up the data, I didn't have to resize partitions, I didn't have to restore data from backup as I would have if I needed to increase the size of a traditional partition that would have been mounted at /home.

LVM Components 3:46-5:56

Basically, LVM makes life a lot easier for the system administrator. With this in mind, let's take a look at the various components that are required to create these logical volumes we've been talking about. Understand that LVM creates a pool of storage space called a volume group.

From the space contained in the volume group we can allocate logical volumes. We define the volume and we assign them a certain amount of space. The structure of LVM uses the components that you see here.

First of all, we have our physical volumes. Physical volume could be one of two different things: it could be a partition on a hard disk drive or it could even be an entire hard disk drive--all the space on the drive. Either way works.

The key thing to remember is that the storage space available on these physical volumes is then assigned and pooled together into the volume group. This is where things get really cool with LVM, because once we assign storage space to the volume group, we are no longer concerned about physical partition or disk boundaries. It's all just available storage space.

Basically, the volume group consists of all the space available on these physical volumes grouped together. The key thing to remember here is the fact that the storage space within the volume group can come from many different physical volumes on many different storage devices.

What this means is you can add additional hard disks or additional partitions to the volume group whenever you decide you need additional storage space. As we said earlier, we take the space in the volume group and we allocate it to individual logical volumes. These logical volumes are really kind of analogous to physical partitions that you might be used to with either GUID or MBR.

You can format a logical volume and add a file system to it. You can create a file system on a logical volume, such as ext3, ext4, riser, whatever it is you want to use. And then you can then mount the logical volume in a directory in your file system and store data on it, just like you would a partition.

Create LVM Volumes 5:56-6:48

The process for creating a logical volume is shown here. The first thing you have to do is create your physical volumes, and then you need to assign those physical volumes to a volume group. Once you've done that, you create your logical volumes out of the volume group. Let's look at the first step in the process, where you define your LVM physical volumes.

It's important at this point that you don't confuse physical volumes with logical volumes. The physical volumes are composed of the disk partitions--or even the entire hard disk drive, if you want to--that we defined as physical LVM volumes that are then allocated to a volume group.

If you decide to use an existing partition as a physical volume for your volume group, then you need to set its partition type to Linux LVM, which I believe the type code is 8e.

Clear a Disk's Partition Table 6:45-8:06

I have tried it with just standard Linux partitions, type 83, and it works just fine. Most of the time when I set up LVM, I actually use partitions, but if you want to, you could use the entire hard disk itself without creating any partitions on it. You can use the entire hard disk as a physical volume, but be aware that if you decide to do this, the hard disk cannot contain a partition table.

If you've ever created a partition ever, at all, on that hard disk drive, then it's going to have a partition table, even if there are no partitions in it. If the disk you want to use already does have an existing partition table, you need to clear it out and you can use the dd command, as shown here, to do just that.

In this example, we're going to clear out the partition table for an MBR disk. To do this, we just have to override the first 512 KB on the hard disk drive because this is where the master boot record resides.

In this example, we're going to take just junk characters from the /dev/0 file and we're going to write them to the hard disk itself /dev/sdb. We're going to copy 512 KB and we're only going to do it once. So we're essentially overriding the first 512 K block on the hard disk drive with nothing.

Define LVM Physical Volumes 8:07-9:01

Once you've decided which disks or partitions you want to use, you then use the pvcreate command at the shell prompt to define these disks or partitions as physical volumes. The syntax is really easy. You just type pvcreate, followed by the device filename of the device that you want to add. In this case, we are adding an entire hard disk drive.

First I define sdb as a physical volume, then I define sdc as a physical volume, and then I define sdd as a physical volume. Once you've done that, you can use the pvscan -v command to view all the physical volumes that are currently defined on the system, along with their size.

Here you can see that I have sdd, sdc, and sdb. It also shows their size over here. Notice they're not the same size--sdd and sdc are 16 GB in size, sdb is 20 GB in size--and LVM does not care.

Create a Volume Group 9:02-10:24

Once we've got our physical volumes defined, we then need to assign those physical volumes to the volume group. This basically takes the space on those physical volumes and allocates it to the volume group that can then be later on assigned to a logical volume.

To create a volume group, you use the vgcreate command. The process for allocating physical volumes to volume group is a little bit tricky if the volume group doesn't already exist.

In order to create the volume group, we first specify the vgcreate command, followed by the name of the volume group we want to create, and then the name of the first physical volume that we want to assign to that volume group.

In this case, I'm creating a volume group named RESEARCH and then I'm assigning the sdb physical volume to that volume group. In this case, I want to add two more physical volumes to the group as well, but I could not do that from the vgcreate command. On some distributions you can, but most distributions you cannot. You can specify only one physical volume with vgcreate command.

Therefore, what you have to do is run a second command called vgextend, followed by the volume name, and then you can list out the rest of the physical volumes that you want to add to the volume group. Run the pvscan command, and then you can see that sdb, c, and d are all added to the same volume group called research.

Define Logical Volumes 10:23-11:57

Once we have our volume group defined, we can then use the space that we've allocated to that volume group to define our logical volumes, our LVs. To do this, you use the lvcreate command, the syntax is shown here.

You enter lvcreate then use the dash L (-L) option to specify the size of the volume that you want to create, then you use the dash n (-n) option to specify the name of the volume, and then at the end of the command you specify which volume group you want to steal that space from to create the volume in question.

In this case, I am creating a 10 GB volume named res_vol, and the space is going to come from the RESEARCH volume group. I did the same thing a second time to create a second volume within that same volume group named dev_vol.

This is really cool. Essentially what we just did is define two logical volumes within a single volume group, which itself was created by pooling together all of the storage space from the three hard disk drives.

The cool thing about it is if I ended up needing more space for any one of these volumes, dev_vol or res_vol, all I would have to do is install a new storage device in the system, define it as a physical volume, allocate a space to the volume group, and then run the lvextend command to increase the size of the volume.

I didn't have to back up any data, I didn't have to resize any partitions; I just dynamically add the new storage space and the volume group to the appropriate logical volume.

Mount Logical Volumes 11:56-13:10

Once we have the logical volumes created, we then need to treat them basically like we would an MBR or GPT partition. In order to store data on an MBR or GPT partition, I have to create a file system on it and then mount it in the file system, and the same holds true for logical volumes.

The first thing I need to do is create a file system on it, just like I would a partition, and use the exact same command to do so. I enter mkfs -t, followed by the type of file system that I want to create.

Then instead of specifying a partition that I want to create the file system on, I instead specify the device file for the logical volume, which uses a slightly different syntax than a partition. We enter /dev/, followed by the name of the volume group that the logical volume resides in, and then the name of the logical volume itself.

Once the file system is created on the logical volume, I can mount it in the file system. And I use the mount command, just like I would with a MBR or GPT disk partition, and I specify -t followed by the type of file system that I just created on the logical volume.

And then once again I specify the device file for the logical volume using the exact same syntax that I did before. And then I end the command with where I want to mount it in the file system.

Summary 13:11-13:24

That's it for this lesson. In this lesson we talked about how to use LVM on a Linux system. We first talked about how LVM works, we talked about the components that are required within an LVM implementation, and then we ended this lesson by reviewing the process for creating a logical LVM volume.
