Quick Guides
Common LVM Recipes

Overview

LVM (Logical Volume Manager) allows you to take a portion of a block level device and use it as a repository for other block level devices. The most common example of a "block level device" is a hard drive, but it can also include a RAID array, a file, anything that you would normally access as you access the standard hard drive in your home computer. USB thumb drives, etc...

Basically, you set aside a block of storage and let it be controlled by LVM instead of the normal hard drive access. The block of storage can be the entire disk, a partition, or even an LVM block (yes, you can have LVM inside of LVM, and it is quite useful at times).

In the following example, we'll take a RAID set (named md0) and use the entire device as an LVM pv "Physical Volume".

pvcreate /dev/md0

At this point, you can create a vg (Volume Group) using the pv. We'll call it "vg0".

lvcreate vg0 /dev/md0

Now, you have a volume group "vg0" which has all of the storage available on the physical volume /dev/md0 (well, with a little overhead subtracted). Now, let's say you want to allocate 10G of storage for some testing. We'll call it "testing".

lvcreate -L 10G -n testing vg0

At this point, testing is an **unformated**, **unpartitioned** block level device (hard drive). So, you can do anything you would normally do with it.

fdisk /dev/vg0/testing

mkfs.ext4 -m 0 -L testing /dev/vg0/testing

Reduce the size of an Logical Volume (LV) with ext2/3/4 under it

Before you can reduce a partition, you must reduce the size of the underlying file system. I'm showing ext2/3/4 (tested on ext4) where you use the command resize2fs to do this. Obviously, if you do not care about the underlying file system, you can simply reduce the size and then format it as you like. So, for swap, simply umount the swap (swapoff), resize (lvreduce), recreate the swap (mkswap), then mount it (swapon)

For most file systems, you must unmount it first. If you're wanting to resize /, you'll need to boot from a CD to do this. Now, do your work.

```
# umount the file system first! Very important
#
# You MUST do an fsck on the file system
e2fsck -f /dev/vg0/partition
# resize the file system to the minimum size possible
# The -p means "show me feedback" and the -M means "minimum"
resize2fs -pM /dev/vg0/partition
# now, reduce the actual partition size.
# It MUST be larger than the file system size
# the -L ###G tells it what the new size will be
lvreduce -L 200G /dev/vg0/partition
# grow the file system to the maximum of the partition
resize2fs /dev/vg0/partition
# check the file system again
e2fsck -f /dev/vg0/partition
```

You can now remount the system.

This was based on an article at https://blog.shadypixel.com/how-to-shrink-an-lvm-volume-safely/

Increase the size of a Logical Volume (LV) with ext2/3/4 under it

Sometimes you need more space in the logical volume you create. In this case, you must unmount the volume, grow the partition (LV) size, then grow the file system under it. This is the reverse of the above section.

We'll just use resize2fs to grow the file system, and lvextend to grow the partition

```
# umount the file system first! Very important
#
# You MUST do an fsck on the file system
e2fsck -f /dev/vg0/partition
# now, increase the actual partition size.
# the -L ###G tells it what the new size will be
lvextend -L 200G /dev/vg0/partition
# grow the file system to the maximum of the partition
resize2fs /dev/vg0/partition
# check the file system again
e2fsck -f /dev/vg0/partition
```

Note: lvextend allows you some shorthand to grow, based on the size of the Volume Group (VG) or the amount that is available. In this case, you use the lower case "L" for the size parameter, and use the below syntax for the size you created:

+###% followed by the key **VG**, **LV**, **PVS**, **FREE** or **ORIGIN**

Thus, if we wanted to extend the LV to 75% of the amount of free space on the VG, you would say:

```
lvextend -l +75%FREE /dev/vg0/partition
```

According to the documentation (man page) using lvextend without the -L or -l parameters is equivilent to:

```
lvextend /dev/vg0/partition /dev/device_name
# same as
lvextend -l +100%PVS /dev/vg0/partition
```

meaning it will try to take the entire size of the Physical Volume (PV) (/dev/devicename in this case). Obviously, this won't work if you already have an LV on it.

Testing a change with ability to revert

Problem: You are using LVM as an underlying structure. You want to make changes, but are not sure if the changes will cause problems. Solution is to create a snapshot of the LV, make your modifications, then revert if they cause problems.

In this, I will assume volume group vg0 has a logical volume virt has a size of 10G, and the changes will probably only be about 1 Gig. What that contains is irrelevant.

1. Shut down any processes accessing the logical volume. This is not required, but, for example, if you create the snapshot of a partion on a running virtual, it is the equivilent of turning the power off on a running program; possible data loss if you recover.
2. Create snapshot with the command

   ```
   lvcreate -s -L 1G -n snap.virt /dev/vg0/virt
   ```

   1. The size of only a gig is useful to keep the amount of space needed to a minimum. You can, of course, make the snapshot the same as the original. The caveat is, **the size of the snapshot must be able to contain all changes in the original** system
   2. I gave the snapshot a name (-n parameter). If you do not do this, a name will be generated by LVM
   3. You can view how "full" the snapshot is with the command

```
lvs
```

3. Make your changes. For example, if your snapshot is of a Xen virtual, start the virtual back up and make the system changes.
4. Clean up.
    1. If you have problems with the existing system, revert to the original
        1. Shut down any processes accessing the original volume (ie, shut down a xen virtual, or unmount a partition).
        2. Revert the original with the command

           ```
           lvconvert --merge /dev/vg0/snap.virt
           ```

           The parameter is the path to the snapshot. This will take all changes to the original and put them back in, then it will automatically delete the snapshot.
    2. If you had no problems and want the machine in the new state permenantly, remove the snapshot with

       ```
       lvremove /dev/vg0/snap.virt
       ```

       be sure you don't delete the wrong one, that is why I preceed snapshots with "snap"
5. Please note: running an LV with a snapshot decreases efficiency. Any writes to the original generate a write to the snapshot, so you are decreasing disk access speed greatly. Don't leave spare snapshots laying around past the time you need them.

Enlarging Physical Volume

Problem: You are using LVM as an underlying file structure on a virtual, and you need more room. Solution is to simply grow the underlying container, then use pvresize to let lvm2 know about it.

System

- Xen hvm
- Physical volume for DOMU (hvm) is an LVM partition exported as a device (full disk). We'll call this /dev/virtuals/my_domu-disk1 in the following example. This is from the DOM0's perspective. It is 50G and we want it to become 100G, with all that space available to the DOMU.
- One or more partitions on the exported device has been set as a pv (Physical volume) from within the DOM. Yes, we're talking about an LVM running on top of an LVM. We'll call this /dev/xvdb1, and it is from the DOMU's perspective.
- The partition used for the pv is the last one in the exported device. If this is NOT the case, stop now and figure out something else.

Solution

- Shut down the DOMU
- lvresize -L 100G /dev/virtuals/my_domu-disk1
- fdisk /dev/virtuals/my_domu-disk1
    - use the "p" command to see what the partition number and type is and, most importantly, what its starting cylinder/sector/whatever is. You **must** record this info. If it is bootable (unlikely), you will need that also.
    - use the "d" command to delete the partition.
    - use the "n" command to create a new partition.
        - Use the same partition number
        - Use the same starting cylinder/sector/whatever as the one you just deleted.
        - Let the ending cylinder/whatver be the default, ie the rest of the available space
    - use the "p" command again to ensure the partition is an exact duplicate of what it was, except the ending location/size should now be greater
    - use the "w" command to write the changes to disk. You will most likely get an error that the partition table has not been updated in the kernel, but you can safely ignore that. If you don't want to ignore it, run the following command to force a reread, but since we don't care if the DOM0 knows about the change, I generally ignore this.
        - kpartx -lv /dev/virtuals/my_domu-disk
- Start the DOMU and log into it
- vgs # so we can see what we had
- pvresize /dev/xvdb1
- vgs # you should now see the additional 50G available to your DOMU

Swapping Physical Volumes

Common LVM Recipes Problem: You want to swap out an existing Physical Volume for a new one. Maybe a faster drive, or a better RAID set. To do this is a multi-step process, but it can be done with no downtime, though it will definitely put a strain on the drive subsystem, especially if you are moving from one software RAID set to another.

Following example assumes you have an existing Volume Group that consists of one hardware RAID partition (sdb1), and you want to move it to a software RAID partition (md0). I assume you know how to build the RAID array. To replace /dev/sdb1 with /dev/md0, you must first add md0 to the volume group, then move all data from the volume group to it, then remove sdb1 from the volume group. Note: the pvmove must move all allocated extents from sdb1 to md0 **without** having any downtime on the server. Thus, it takes a long time (hours, maybe days). However, if it dies in the middle, you can simply restart it with the same command.

```
# First, set a device to become a new physical volume. NOTE: This must
 be capable of holding all USED data in the current vg (ie, sum of all
```

```
 lv's)
pvcreate /dev/md0
# Now, add it to the volume group (in this case, datalvm)
vgextend /dev/datalvm /dev/md0
# And move all data from the old pv to the new one
pvmove -v /dev/sdb1
# once done (and it can take hours), remove the old one from the volum
e group
vgreduce datalvm /dev/sdb1
# and, if you want, remove the PV flag from the volume.
pvremove /dev/sdb1
```

Unique solution ID: #1009
Author: Rod
Last update: 2016-06-27 00:18