Anthony's Blog
Thoughts, ideas, and ramblings on Linux, electronics and science

LVM Loopback HOW-TO

Posted on December 19, 2010

This is a simple tutorial on setting up LVM on loopback devices, I've used it a few times for creating dynamic virtual disks; it came in particularly handy when archiving NEXRAD radar data for my radarwatchd project – using up all your inodes on several hundreds of thousands of 15Kb files doesn't sound like my idea of fun. Creating a virtual volume with reiserfs was a particularly handy solution in this case.

First Steps

Create several empty files with dd. These will hold your physical volumes:

```
# dd if=/dev/zero of=lvmtest0.img bs=100 count=1M
1048576+0 records in
1048576+0 records out
104857600 bytes (105 MB) copied, 8.69891 s, 12.1 MB/s
# dd if=/dev/zero of=lvmtest1.img bs=100 count=1M
1048576+0 records in
1048576+0 records out
104857600 bytes (105 MB) copied, 8.69891 s, 12.1 MB/s
```

Link them to loopback devices with losetup: (see below if you run out of loopback devices)

```
# losetup /dev/loop0 lvmtest0.img
# losetup /dev/loop1 lvmtest1.img
```

Partition them with fdisk. Create a single primary partition, full size of the device and set the type to Linux LVM (0x8e). Shorthand commands with fdisk: n,p,1,Enter,Enter,t,8e,w. I've also automated this somewhat with sfdisk:

```
# sfdisk /dev/loop0 << EOF
,,8e,,
EOF
```

Install and configure LVM if needed. In this test I used the filter settings in lvm.conf to ensure only loopback devices will be used with LVM:

```
filter = [ "a|/dev/loop.*|", "r/.*/" ]
```

Initialize LVM:

```
# vgscan
Reading all physical volumes.  This may take a while...
# vgchange -a
```

Create physical volumes with pvcreate:

```
# pvcreate /dev/loop0 /dev/loop1
 Physical volume "/dev/loop0" successfully created
 Physical volume "/dev/loop1" successfully created
```

Create a volume group then extend to include the second and any subsequent physical volumes:

```
# vgcreate testvg /dev/loop0
 Volume group "testvg" successfully created
# vgextend testvg /dev/loop1
 Volume group "testvg" successfully extended
```

Verify the operation was successful:

```
# vgdisplay -v
 Finding all volume groups
 Finding volume group "testvg"
 --- Volume group ---
 VG Name               testvg
 System ID
 Format                lvm2
 Metadata Areas        2
 Metadata Sequence No  2
 VG Access             read/write
 VG Status             resizable
 MAX LV                0
 Cur LV                0
 Open LV               0
 Max PV                0
 Cur PV                2
 Act PV                2
 VG Size               192.00 MB
```

```
 PE Size               4.00 MB
 Total PE              48
 Alloc PE / Size       0 / 0
 Free  PE / Size       48 / 192.00 MB
 VG UUID               1Gmt3W-ivMH-mXQH-HswP-tjHR-9mAZ-917z0g

 --- Physical volumes ---
 PV Name               /dev/loop0
 PV UUID               X11MYK-u8hk-4R26-CHuy-QUSw-2hLq-Notlnc
 PV Status             allocatable
 Total PE / Free PE    24 / 24

 PV Name               /dev/loop1
 PV UUID               dLKXlz-c536-9Elj-C2zZ-B4aw-69kj-zZ7PuN
 PV Status             allocatable
 Total PE / Free PE    24 / 24
```

Create a logical volume. Use the largest available size reported to use by vgdisplay (Free PE/Size value):

```
# lvcreate -L192M -ntest testvg
 Logical volume "test" created
```

Finally create a filesystem on the new logical volume. We're using reiserfs for this example here:

```
# mkfs.reiserfs /dev/mapper/testvg-test
mkfs.reiserfs 3.6.21 (2009 www.namesys.com)

A pair of credits:
Alexander Lyamin keeps our hardware running, and was very generous to our
project in many little ways.

The Defense Advanced Research Projects Agency (DARPA, www.darpa.mil) is the
primary sponsor of Reiser4. DARPA does not endorse this project; it merely
sponsors it.

Guessing about desired format.. Kernel 2.6.31-17-generic-pae is running.
Format 3.6 with standard journal
Count of blocks on the device: 49152
Number of blocks consumed by mkreiserfs formatting process: 8213
Blocksize: 4096
Hash function used to sort names: "r5"
Journal Size 8193 blocks (first block 18)
Journal Max transaction length 1024
inode generation number: 0
UUID: d50559af-5078-4de5-812a-264590e60177
ATTENTION: YOU SHOULD REBOOT AFTER FDISK!
 ALL DATA WILL BE LOST ON '/dev/mapper/testvg-test'!
```

```
Continue (y/n):y
Initializing journal - 0%....20%....40%....60%....80%....100%
Syncing..ok
ReiserFS is successfully created on /dev/mapper/testvg-test.
```

Mount the volume and you should be done:

```
# mount /dev/mapper/testvg-test /mnt/virtlvm
# df
Filesystem           1K-blocks      Used Available Use% Mounted on

/dev/mapper/testvg-test
 196596     32840    163756  17% /mnt/virtlvm
```

Growing the LVM volume

To expand the LVM volume you will follow similar steps to the ones stated above:

Create another virtual disk and partition with dd and fdisk:

```
# dd if=/dev/zero of=lvmtest2.img bs=100 count=1M
# fdisk /dev/loop2
```

Tie new disk image to a free loopback device with losetup:

```
# losetup /dev/loop2 lvmtest2.img
```

Create physical volumes on the new device with pvcreate:

```
# pvcreate /dev/loop2
Physical volume "/dev/loop2" successfully created
```

Extend the volume group:

```
# vgextend testvg /dev/loop2
 Volume group "testvg" successfully extended
```

Extend the logical volume:

```
# lvextend /dev/mapper/testvg-test /dev/loop2
 Extending logical volume test to 288.00 MB
 Logical volume test successfully resized
```

Resize the filesystem with resize_reiserfs:

```
# resize_reiserfs /dev/mapper/testvg-test
resize_reiserfs 3.6.21 (2009 www.namesys.com)

resize_reiserfs: On-line resizing finished successfully.

# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/testvg-test
                        294896     32840    262056  12% /mnt/virtlvm
```

Modifying the amount of loopback devices

To see all the loopback devices:

```
# ls -l /dev/loop*
```

Adding new loopback devices
If loopback support is compiled as a module:

```
# modprobe loop max_loop=64
```

To make permanent edit /etc/modules or /etc/modprobe.conf and use add one of the following:

```
loop max_loop=64
options loop max_loop=64
```

If loopback module is compiled into the kernel you will need to reboot, edit the kernel command-line parameters appending 'max_loop=64' to the end.
Loopback devices can also be dynamically created with mknod. Loopback devices have a major number of 7 and minor number of the loopback device. Keep in mind devices created this way will not be persistent and will disappear after a reboot:

```
# mknod -m660 /dev/loop8 b 7 8
# mknod -m660 /dev/loop9 b 7 9
# mknod -m660 /dev/loop10 b 7 10
```

That should cover mostly everything. I've been informed that some of the steps may not be strictly necessary, such as the vgscan/vgchange. I also believe the partitioning may not be needed as we're using the full size of the virtual devices but it's good practice nonetheless and definitely makes things clearer being able to see the LVM partitions. Hope this helps!

If you enjoyed this post, make sure you subscribe to my RSS feed!

This entry was posted in howto, linux, sysadmin and tagged howto, linux, lvm, sysadmin by Anthony DeChiaro. Bookmark the permalink [http://www.anthonyldechiaro.com/blog/2010/12/19/lvm-loopback-how-to/] .

3 Replies
3 Comments
0 Tweets
0 Facebook
0 Pingbacks

Last reply was January 22, 2014

1.  *John*
   View August 7, 2012
   Thanks for that. It helped me get a loop device installed on an OpenFiler box.
   Using it via iSCSI on Windows as I type.
   Appreciated.

   Reply
2.  *Oliver*
   View March 29, 2013
   Thank you for this post. It is a great tutorial. I used it to setup my lvm!

   Reply
3.  *KK*
   View January 22, 2014
   This is excellent, thank you very much!
   Is there any way of making the loop devices and partitions persistent across reboots, apart from scripting the above steps into init/rc scripts?
   Thanks!

   Reply