

# CHAPTER 7

## Managing Storage

Managing storage is an important responsibility. The right solution works seamlessly with little gratitude. The wrong solution can lead to many headaches and late nights of trying to recover from failed file systems or inadequate storage allocation.

During installation, you are asked which partitioning method to use. You must choose to remove Linux partitions on selected drives and create the default layout, remove all partitions on selected drives and create the default layout, use free space on selected drives and create the default layout, or create a custom layout. If you choose to create the default layout, the Logical Volume Manager (LVM) is used to divide the hard drive, and then the necessary Linux mount points are created. Alternatively, if you choose custom layout, you can instead use software RAID or create partitions directly on the hard drives. Global File Systems (GFS) and clustering are two more storage solutions available with Red Hat Enterprise Linux.

This chapter explains these partitioning options so you can determine which is best for you and you can learn how to manage them. It also discusses how to use access control lists to limit access to filesystems as well as how to enforce disk usage limits known as quotas. Analyze how your company uses storage and decide which options are best for you.

### Understanding Partitioning

LVM and RAID offer benefits such as resizing, striping, and combining multiple hard drives into logical physical devices. Sometimes it is necessary to just create partitions on the hard drives. Even when using RAID, partitions are created before the LVM or RAID layer is implemented.

#### IN THIS CHAPTER

- ▶ Understanding Partitioning
- ▶ Understanding LVM
- ▶ Understanding RAID
- ▶ Understanding Clustering and GFS
- ▶ Using Access Control Lists
- ▶ Using Disk Quotas

To view a list of partitions on the system, use the `fdisk -l` command as root. As you can see from Listing 7.1, the output shows each partition along with its device name, whether it is a bootable partition, the starting cylinder, the ending cylinder, the number of blocks, the filesystem identification number used by `fdisk`, and the filesystem type.

LISTING 7.1 Partitioning Scheme with Standard Partitions

---

```
Disk /dev/sda: 100.0 GB, 100030242816 bytes
255 heads, 63 sectors/track, 12161 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	1	1147	9213246	83	Linux
/dev/sda2		1148	4334	25599577+	83	Linux
/dev/sda3		4335	4399	522112+	82	Linux swap / Solaris
/dev/sda4		4400	12161	62348265	5	Extended
/dev/sda5		4400	12161	62348233+	83	Linux

---

If the system uses LVM or RAID, the `fdisk -l` output will reflect it. For example, Listing 7.2 shows the output for a system partitioned with LVM. There are fewer partitions shown because the logical volumes are inside the logical volume group. The first partition shown is the `/boot` partition because it can't be inside a logical volume group.

LISTING 7.2 Partitioning Scheme with LVM

---

```
Disk /dev/sda: 300.0 GB, 300090728448 bytes
255 heads, 63 sectors/track, 36483 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1		1	13	104391	83	Linux
/dev/sda2		14	36482	292937242+	8e	Linux LVM

---

During installation, the hard drives can be partitioned, given a filesystem type for formatting, and assigned a mount point as described in Chapter 1, “Installing Red Hat Enterprise Linux.” If hard drives are added to the system after installation or a hard drive has to be replaced, it is important to understand how to perform these functions post-installation.

### CAUTION

Perform all these actions in rescue mode without the filesystem mounted or ensure the entire device is not mounted before manipulating the partition table for it. Refer to Chapter 10, “Techniques for Backup and Recovery,” for instructions on booting into rescue mode. Most changes to the partition table require a reboot. When you exit rescue mode, the system will reboot.

---

## Creating Partitions

A partition can be created from free space on a hard drive. You might need to create a partition if you add a new hard drive to the system, if you left unpartitioned space on the system during installation and want to partition it, or if you are using LVM and want to create the physical volumes on a partition instead of an entire raw device.

There are two partitioning utilities in Red Hat Enterprise Linux: `parted` and `fdisk`. The `parted` utility is used in this chapter because it includes a resize utility and is a bit more user-friendly. For more information on `fdisk`, refer to the man page with the `man fdisk` command.

As root, issue the `parted` command followed by the device name such as

```
parted /dev/sda
```

You are now in an interactive `parted` shell, in which the commands executed manipulate the device specified. To view existing partitions from this interactive shell, type the `print` command at the (`parted`) prompt. The output should look similar to Listing 7.3. If you compare this output to the output in Listing 7.1 and Listing 7.2 from the `fdisk -l` command, you will see that the `parted` output is a little easier to read because it includes the size in user-friendly units such as megabytes and gigabytes instead of the beginning and ending cylinders from the `fdisk -l` output.

LISTING 7.3 Partition Table from `parted` for Standard Partitions

---

```
Using /dev/hda
(parted) print
Disk geometry for /dev/hda: 0kB - 100GB
Disk label type: msdos
```

Number	Start	End	Size	Type	File system	Flags
1	32kB	9434MB	9434MB	primary	ext3	boot
2	9434MB	36GB	26GB	primary	ext3	
3	36GB	36GB	535MB	primary	linux-swap	
4	36GB	100GB	64GB	extended		
5	36GB	100GB	64GB	logical	ext3	

---

Once again, the output will differ depending on the partitioning scheme being used. Listing 7.4 shows output from a system using LVM and can be compared to Listing 7.2, which shows the same output from `fdisk -l`.

LISTING 7.4 Partition Table from `parted` for LVM

---

```
Disk /dev/sda: 300GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
```

Number	Start	End	Size	Type	File system	Flags
1	32.3kB	107MB	107MB	primary	ext3	
3	107MB	300GB	299.9GB	primary		lvm

---

To create a partition in parted, issue the following command at the interactive parted prompt:

```
mkpart <part-type> <fs-type> <start> <end>
```

<part-type> must be one of primary, logical, or extended. <fs-type> must be one of fat16, fat32, ext2, HFS, linux-swaps, NTFS, reiserfs, or ufs. The <start> and <end> values should be given in megabytes and must be given as integers.

The ext3 filesystem is the default filesystem for Red Hat Enterprise Linux. It is the ext2 filesystem plus journaling. To create an ext3 filesystem, use ext2 as the <fs-type> and then use the -j option to mke2fs to make the filesystem ext3 as described in the next section.

After creating the partition, use the print command again to verify that the partition was created. Then type quit to exit parted.

## Creating a Filesystem on a Partition

Next, create a filesystem on the partition. To create an ext3 filesystem (default used during installation), as root, execute the following, where <device> is the device name for the partition such as /dev/sda1:

```
mke2fs -j <device>
```

If the partition is to be a swap partition, format it with the following command as root:

```
mkswap <device>
```

## Labeling the Partition

To label the partition, execute the following as root:

```
e2label <device> <label>
```

While labeling is not required, partition labels can be useful. For example, when adding the partition to /etc/fstab, the label can be listed instead of the partition device name. This proves useful if the partition number is changed from repartitioning the drive or if the partition is moved.

If the e2label command is used with just the partition device name as an argument, the current label for the partition is displayed.

## Creating a Mount Point

Now that the partition is created and has a filesystem, as root, create a directory so it can be mounted:

```
mkdir <dir-name>
```

Then, mount the new partition:

```
mount <device> <dir-name>
```

such as:

```
mount /dev/sda5 /tmp
```

Access the directory and make sure you can read and write to it.

Finally, add the partition to the `/etc/fstab` file so it is mounted automatically at boot time. For example:

```
LABEL=/tmp          /tmp          ext3    defaults    1 2
```

If a new swap partition is added, be sure to use `swap` as the filesystem type instead:

```
LABEL=swap2        swap          swap    defaults    0 0
```

## Resizing Partitions

The `parted` utility can also be used to resize a partition. After starting `parted` as root on the desired device, use the following command to resize a specific partition:

```
resize <minor-num> <start> <end>
```

To determine the `<minor-num>` for the partition, look at the partition table with the `print` command such as the output shown in Listing 7.3 and Listing 7.4. The `<start>` and `<end>` values should be the start and end points of the partition, in megabytes.

## Removing Partitions

To use `parted` to remove a partition, start `parted` on the desired device as root, and issue the following command at the interactive prompt:

```
rm <minor-num>
```

The minor number for the partition is displayed when you execute the `print` command to list partitions. The data on the partition will no longer be accessible after the partition is removed, so be sure to back up any data you want to keep before removing the partition.

# Understanding LVM

Logical Volume Manager, or LVM, is a storage management solution that allows administrators to divide hard drive space into *physical volumes (PV)*, which can then be combined into *logical volume groups (VG)*, which are then divided into *logical volumes (LV)* on which the filesystem and mount point are created.

As shown in Figure 7.1, because a logical volume group can include more than one physical volume, a mount point can include more than one physical hard drive, meaning the largest mount point can be larger than the biggest hard drive in the set. These logical volumes can be resized later if more disk space is needed for a particular mount point. After the mount points are created on logical volumes, a filesystem must be created on them.

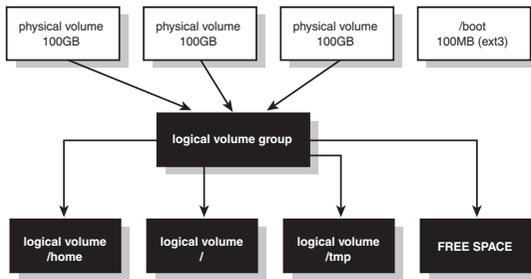


FIGURE 7.1 How Logical Volume Manager Works

LVM is used by default during installation for all mount points except the `/boot` partition, which cannot exist on a logical volume. This section discusses how to perform LVM operations after installation such as creating a physical volume for a newly added hard drive, expanding logical volumes, and generating LV snapshots.

Table 7.1 summarizes the LVM tools available after installation.

TABLE 7.1 LVM Tools

LVM Tool	Description
<code>pvcreate</code>	Create physical volume from a hard drive
<code>vgcreate</code>	Create logical volume group from one or more physical volumes
<code>vgextend</code>	Add a physical volume to an existing volume group
<code>vgreduce</code>	Remove a physical volume from a volume group
<code>lvcreate</code>	Create a logical volume from available space in the volume group
<code>lvextend</code>	Extend the size of a logical volume from free physical extents in the logical volume group
<code>lvremove</code>	Remove a logical volume from a logical volume group, after unmounting it
<code>vgdisplay</code>	Show properties of existing volume group
<code>lvdisplay</code>	Show properties of existing logical volumes
<code>pvscan</code>	Show properties of existing physical volumes

## Adding Additional Disk Space

One big advantage of using LVM is that the size of a logical volume can be increased and logical volumes can be added to create additional mount points. To modify the LVM configuration post-installation, the `lvmd` package needs to be installed. Refer to Chapter 3, “Operating System Updates,” for details on installing packages.

### TIP

If possible, leave free disk space when partitioning during installation so logical volume sizes can be increased without adding additional hard drives.

To increase the size of an existing logical volume or to add a logical volume, you first need free disk space. This free disk space can either be disk space that already exists in the system as unpartitioned space (not part of an existing logical volume), an unused partition, physical volume that is not already a member of a logical volume, or disk space as a result of installing one or more additional hard drives to the system. The disk space can come from removing a logical volume to create space in the logical volume group, however, this is not common because if the LV already exists, it is most likely already being used and cannot be easily deleted without losing data.

After deciding which free disk space to use, the basic steps for increasing the size of a logical volume are as follows:

1. Create new physical volume from free disk space.
2. Add physical volume to the logical volume group.
3. Expand the size of the logical volume to include the newly added disk space in the volume group.
4. Expand the filesystem on the logical volume to include the new space.

To add a logical volume, use the following steps:

1. Create new physical volume from free disk space.
2. Add physical volume to the logical volume group.
3. Create a logical volume with the new space in volume group.
4. Create a filesystem on the logical volume.
5. Create a mount point.
6. Mount the logical volume.
7. Test the filesystem.
8. Add the new mount point to `/etc/fstab`.

#### TIP

If you prefer a graphical interface, the `system-config-lvm` utility can be used to modify your LVM configuration.

## Creating a Physical Volume

To create a new physical volume from free hard drive space or a hard drive partition, use the `pvcreate` command:

```
pvcreate <disk>
```

Replace `<disk>` with the device name of the hard drive:

```
pvcreate /dev/sda
```

or the partition name:

```
pvccreate /dev/sda1
```

The <disk> specified can also be a meta device or loopback device, but using an entire hard disk or partition is more common. After creating a physical volume, you can either add it to an existing volume group or create a new volume group with the physical volume.

## Creating and Modifying Volume Groups

A volume group can be created from one or more physical volumes. To scan the system for all physical volumes, use the `pvscan` command as root. It displays all PVs on the system. If the PV is part of a VG, it will display the name of the VG next to it.

To create a VG, execute the `vgcreate` command as root, where <vgname> is a unique name for the volume group and <pvlist> is one or more physical volumes to use, each separated by a space:

```
vgcreate <vgname> <pvlist>
```

For example, to create a VG with the name `DatabaseVG` from the first and second SCSI hard drives:

```
vgcreate DatabaseVG /dev/sda /dev/sdb
```

### NOTE

If the volume group was created during installation, the installation program names the first volume group `VolGroup00`, the second one `VolGroup01`, and so on.

If a volume group already exists but needs to be expanded, use the `vgextend` command to add additional physical volumes to it:

```
vgextend <vgname> <pvlist>
```

To remove a physical volume from a volume group:

```
vgreduce <vgname> <pvlist>
```

Use caution when reducing a volume group because any logical volume using the PVs are removed from the VG and can no longer be accessed.

## Creating and Modifying Logical Volumes

Now that the physical volumes are formed into volume groups, the volume groups can be divided into logical volumes, and the logical volumes can be formatted with a filesystem and assigned mount points.

Use the `lvcreate` command to create a logical volume. Each LV must have a unique name. If one is not specified with the `-n <name>` option, a name will be assigned to it. To create a logical volume from the volume group `<vgname>` of a certain size, specify the size unit after the value of the size such as `300G` for 300 gigabytes:

```
lvcreate -n <lvname> --size <size> <vgname>
```

Each physical volume consists of *physical extents*, which are 4 megabytes in size by default. When the size is given in gigabytes, this size must be converted to physical extents, meaning that some amount of disk space may not be used. So, the number of physical extents to use when creating the logical volume can be given with the `-l <numpe>` option:

```
lvcreate -n <lvname> -l <numpe> <vgname>
```

To determine the number of physical extents in a logical volume group, issue the following command as root:

```
vgdisplay <vgname>
```

The `Total PE` line shows the number of physical extents for the volume group. The output should look similar to Listing 7.5, which shows a total of 1189 physical extents. Look for the `Free PE / Size` line to determine whether any free PEs are available to allocate to a new logical volume. Listing 7.5 shows 220 free physical extents.

#### LISTING 7.5 Example `vgdisplay` Output

---

```

--- Volume group ---
VG Name           VolGroup00
System ID
Format            lvm2
Metadata Areas    1
Metadata Sequence No 5
VG Access          read/write
VG Status          resizable
MAX LV            0
Cur LV           2
Open LV           2
Max PV            0
Cur PV           1
Act PV            1
VG Size           37.16 GB
PE Size           32.00 MB
Total PE          1189
Alloc PE / Size   969 / 30.28 GB
Free PE / Size    220 / 6.88 GB
VG UUID           N60y5U-2sM2-uxHY-M1op-Q1v3-uVV2-Zkahza

```

---

**TIP**

Each LV has a device name in `/dev/` with the format `/dev/<vgname>/<lvname>`.

By default, logical volumes are created linearly over the physical volumes. However, they can be striped over multiple PVs:

```
lvcreate -i<stripes> -I<stripesize> -l <numpe> -n <lvname> <vgname> <pvlist>
```

The `-i<stripes>` option sets the number of stripes, or physical volumes to use. The `-I<stripesize>` is the stripe size, which must be  $2^n$ , where  $n$  is an integer from 2 to 9. Provide the number of PEs to use with the `-l <numpe>` option or give the size of the LV with the `--size <size>` option. The `-n <lvname>` option specifies the LV name, and `<vgname>` represents the name of the VG to use. Optionally, list the PVs to use, `<pvlist>`, at the end of the command separated by spaces. The number of PVs listed should be equal to the number of stripes.

After creating the logical volume, you must create a filesystem on it. To create an ext3 filesystem, execute the following as root:

```
mke2fs -j /dev/<vgname>/<lvname>
```

If the LV is to be used as swap, execute the following as root instead:

```
mkswap /dev/<vgname>/<lvname>
```

Next, still as the root user, create an empty directory as its mount point with the `mkdir` command, and use the `mount` command to mount the filesystem:

```
mount /dev/<vgname>/<lvname> /mount/point
```

If it mounts properly, the last step is to add it to `/etc/fstab` so it is mounted automatically at boot time. As root, add a line similar to the following, replacing with the appropriate values:

```
/dev/<vgname>/<lvname> /mount/point          ext3    defaults    1 2
```

To extend a logical volume, expand the volume group if necessary, and then use the `lvextend` command. Either specify the final size of the logical volume:

```
lvextend --size <size> /dev/<vgname>/<lvname>
```

or specify how much to expand the logical volume:

```
lvextend --size +<addsize> /dev/<vgname>/<lvname>
```

Just like physical volumes are composed of 4KB physical extents, logical volumes consist of *logical extents*, which also have a default size of 4KB. Instead of specifying the size or amount of space to add in gigabytes, it is also possible to use the `-l <numle>` to provide

the final number of logical extents or `-1 +<numle>` to expand the logical volume by a certain number of logical extents.

After extending the logical volume, the filesystem on it must be expanded as well. If it is an ext3 filesystem (default filesystem for Red Hat Enterprise Linux), it can be expanded while it is still mounted (also known as *online*). To do so, execute the following as root:

```
resize2fs /dev/<vgname>/<lvname>
```

The filesystem is expanded to fill the entire logical volume unless a size is listed after the logical volume device name (be sure to list the size unit such as G for gigabyte after the size):

```
resize2fs /dev/<vgname>/<lvname> <size>
```

To remove a logical volume from a volume group, first unmount it with the `umount` command:

```
umount /dev/<vgname>/<lvname>
```

and then use the `lvremove` command:

```
lvremove /dev/<vgname>/<lvname>
```

To view the existing logical volumes along with information about them such as what VG they are a member of, the number of logical extents, and their size in gigabytes, execute the `lvdisplay` command as root as shown in Listing 7.6.

#### LISTING 7.6 Example `lvdisplay` Output

```
--- Logical volume ---
LV Name           /dev/VolGroup00/LogVol00
VG Name           VolGroup00
LV UUID           tugMfo-PESp-3INs-nrGF-K0Wh-s3U0-19FsTc
LV Write Access   read/write
LV Status         available
# open           1
LV Size           12.94 GB
Current LE        414
Segments         1
Allocation        inherit
Read ahead sectors 0
Block device      253:0

--- Logical volume ---
LV Name           /dev/VolGroup00/LogVol01
VG Name           VolGroup00
LV UUID           fdKfYP-wIP9-M4Da-eoV3-pP99-w8Vb-0yhgZb
LV Write Access   read/write
```

LISTING 7.6 Continued

---

```

LV Status          available
# open             1
LV Size            78.12 GB
Current LE         2500
Segments           1
Allocation         inherit
Read ahead sectors 0
Block device       253:1

--- Logical volume ---
LV Name            /dev/VolGroup00/LogVol02
VG Name            VolGroup00
LV UUID            bzr4Ag-rDKT-y8zY-F3e8-SaBI-QY51-r61J3T
LV Write Access    read/write
LV Status          available
# open             1
LV Size            1.94 GB
Current LE         62
Segments           1
Allocation         inherit
Read ahead sectors 0
Block device       253:2

```

---

## Creating Snapshots

With LVM, it is possible to take a snapshot of a logical volume while the LV is still in read-write mode and being accessed by the system. As the root user, issue the following command:

```
lvcreate --size <size> -s -n <snapshotname> <lvname>
```

The `lvcreate` command is used to create a new logical volume, meaning there must be free physical extents in the logical volume group to create a snapshot. The `-s` option means that the LV is a snapshot, `<snapshotname>` is the name of the new LV created, and `<lvname>` is the name of the LV from which to create the snapshot.

A snapshot is not a copy of the entire LV. Instead, it keeps track of the changes from the time the snapshot is taken and the present time. Thus, the size of the snapshot LV does not need to be as large as the LV from which it is created. It just needs to be as big as all the changes from the time the snapshot is taken until the snapshot is used. Snapshots are not intended to be left around for long periods of time. Reasons to create snapshots include performing backups (most common), creating virtual machines using the Virtualization feature (refer to Appendix B, “Creating Virtual Machines”), creating a duplicate testing system, and transferring data from one logical volume group (and possibly a different hard drive) to another.

If a snapshot LV reaches disk capacity, it will become unusable. When the backup or data transfer has been completed, the snapshot logical volume should be unmounted and removed with the `lvremove /dev/<vgname>/<lvname>` command. Because the snapshot LV is storing a copy of all changes made to the original LV, performance for the original LV can be reduced because of this copy process.

## Understanding RAID

RAID (Redundant Array of Independent Disks) allows an administrator to form an array of several hard drives into one logical drive recognized as one drive by the operating system. It also spreads the data stored over the array of drives to decrease disk access time and accomplish data redundancy. The data redundancy can be used to recover data should one of the hard drives in the array crash.

There are two types of RAID: *hardware RAID* and *software RAID*. Hardware RAID is implemented through the disk controller for the system. Instructions for configuring hardware RAID differ from controller to controller, so refer to the manual for your disk controller for instructions. Software RAID is implemented through the operating system and does use some processor and memory resources, although some software RAID implementations can produce faster disk access times than hardware RAID.

During installation, it is possible to configure software RAID as discussed in Chapter 1. This section explains the different RAID levels available with software RAID so you can decide which level is best for you. Software RAID allows for RAID levels 0, 1, 5, and 6.

RAID *level 0*, or *striping*, means that data is written across all hard drives in the array to accomplish the fast disk performance. No redundancy is used, so the size of the logical RAID drive is equal to the size of all the hard drives in the array. Because there is no redundancy, recovering data from a hard drive crash is not possible through RAID.

RAID *level 1*, or *mirroring*, means that all data is written to each disk in the array, accomplishing redundancy. The data is “mirrored” on a second drive. This allows for easy recovery should a disk fail. However, it does mean that, for example, if there are two disks in the array, the size for the logical disk is size of the smaller of the two disks because data must be mirrored to the second disk.

RAID *level 5* combines striping and *parity*. Data is written across all disks as in RAID 0, but parity data is also written to one of the disks. Should a hard drive failure occur, this parity data can be used to recover the data from the failed drive, including while the data is being accessed and the drive is still missing from the array.

RAID *level 6* is RAID level 5 with *dual parity*. Data is written across all disks as in RAID 5, but two sets of parity data is calculated. Performance is slightly worse than RAID 5 because the extra parity data must be calculated and written to disk. RAID 5 allows for recovery using the parity data if only one drive in the array fails. Because of the dual parity, RAID 6 allows for recovery from the failure of up to two drives in the array.

## Setting Up RAID Devices

For best results, software RAID should be configured during installation, but it can be configured after installation if necessary. To set up software RAID devices after installation, install the `mdadm` software package. Refer to Chapter 3 for instructions on installing packages. This section provides an overview of post-installation software RAID configuration. It shows you how to create a RAID array and then move the data from the existing filesystem onto it. Be sure to test the process on a test system before attempting it on a production system.

### CAUTION

Remember to back up all data before converting partitions to software RAID devices. As with any process that modifies disk partitions and partition tables, data loss is possible.

Before starting the conversion, add the appropriate number of hard drives with the proper sizes for the RAID level. For example, two partitions are needed for RAID 1 (mirroring) and at least three partitions are needed for RAID 5. To use all the benefits of RAID, each partition in a RAID device should be on separate hard drives so each member of the RAID device can be written to at the same time and there is redundancy across separate hard drives should one fail.

It is possible to configure a RAID array with a missing partition so that the data on the existing partition can be copied to the degraded array. The existing partition is reconfigured as a RAID partition and then added to the RAID array to complete it. However, the process for doing so is more complicated and not recommended because it is easier to lose the existing data. It is recommended that new drives be used to set up the RAID device and for the existing data to then be copied to the new RAID device.

When creating partitions to use for the RAID device, make sure they are of type `Linux raid auto`. In `fdisk`, this is partition id `fd`. After creating the partitions for the RAID device, use the following syntax as the root user to create the RAID device:

```
mdadm --create /dev/mdX --level=<num> --raid-devices=<num> <device list>
```

The progress of the device creation can be monitored with the following command as root:

```
tail -f /proc/mdstat
```

For example, to create a RAID level 1 device `/dev/md0` from three partitions, use the following command:

```
mdadm --create /dev/md0 --level=1 --raid-devices=3 /dev/sda5 /dev/sda6 /dev/sda7
```

The command `cat /proc/mdstat` should show output similar to Listing 7.7.

## LISTING 7.7 Creating a RAID Array

---

```
Personalities : [raid0] [raid1]
md0 : active raid1 sda7[2] sda6[1] sda5[0]
      10241280 blocks [3/3] [UUU]
      [>.....] resync = 0.0% (8192/10241280) finish=62.3min
      speed=2730K/sec

unused devices: <none>
```

---

The RAID device `/dev/md0` is created. Next, create a filesystem on it. To create an ext3 filesystem, execute the following as root:

```
mke2fs -j /dev/md0
```

If the new RAID device is to be used as the swap partition, use the following command as root instead:

```
mkswap /dev/md0
```

Copy any data over to the new device and be sure to change all references to the old partition to the new RAID device, including `/etc/fstab` and `/etc/grub.conf`. It is recommended that the `/boot` and the `/` filesystems remain on their original filesystems to ensure the system can still boot after added the RAID devices. Partitions such as `/home` will benefit from RAID more because data on it changes frequently.

## Adding and Failing RAID Partitions

To add a partition to a RAID device, execute the following as root after creating the partition of type Linux `raid` `auto` (`fd` in `fdisk`):

```
mdadm /dev/mdX -a <device list>
```

To add `/dev/sda8` to the `/dev/md0` RAID device created in the previous section:

```
mdadm /dev/md0 -a /dev/sda8
```

Listing 7.8 shows the output from `cat /proc/mdstat`. The `/dev/sda8` partition is now a spare partition in the RAID array.

## LISTING 7.8 Adding a Spare Partition

---

```
Personalities : [raid0] [raid1]
md0 : active raid1 sda8[3](S) sda7[2] sda6[1] sda5[0]
      10241280 blocks [3/3] [UUU]
      [>.....] resync = 0.6% (66560/10241280) finish=84.0min
      speed=2016K/sec

unused devices: <none>
```

---

If a partition in the array fails, use the following to remove it from the array and rebuild the array using the spare partition already added:

```
mdadm /dev/mdX -f <failed device>
```

For example, to fail `/dev/sda5` from `/dev/md0` and replace it with the spare (assuming the spare has already been added):

```
mdadm /dev/md0 -f /dev/sda5
```

To verify that the device has been failed and that the rebuild has been complete and was successful, monitor the `/proc/mdstat` file (output shown in Listing 7.9):

```
tail -f /proc/mdstat
```

Notice that `/dev/sda5` is now failed and that `/dev/sda8` has changed from a spare to an active partition in the RAID array.

---

#### LISTING 7.9 Failing a Partition and Replacing with a Spare

---

```
Personalities : [raid0] [raid1]
md0 : active raid1 sda8[3] sda7[2] sda6[1] sda5[4](F)
      10241280 blocks [3/2] [_UU]
      [>.....] recovery = 0.2% (30528/10241280) finish=11.1min
      speed=15264K/sec

unused devices: <none>
```

---

## Monitoring RAID Devices

The following commands are useful for monitoring RAID devices:

- ▶ `cat /proc/mdstat`: Shows the status of the RAID devices and the status of any actions being performed on them such as adding a new member or rebuilding the array.
- ▶ `mdadm --query /dev/mdX`: Displays basic data about the device such as size and number of spares such as:

```
/dev/md0: 9.77GiB raid1 3 devices, 1 spare.
```

Add the `--detail` option to display more data (`mdadm --query --detail /dev/mdX`):

```
/dev/md0:
  Version : 00.90.03
  Creation Time : Mon Dec 18 07:39:05 2006
  Raid Level : raid1
  Array Size : 10241280 (9.77 GiB 10.49 GB)
```

```

Device Size : 10241280 (9.77 GiB 10.49 GB)
Raid Devices : 3
Total Devices : 4
Preferred Minor : 0
Persistence : Superblock is persistent

```

```

Update Time : Mon Dec 18 07:40:01 2006
State : clean, degraded, recovering
Active Devices : 2
Working Devices : 3
Failed Devices : 1
Spare Devices : 1

```

```
Rebuild Status : 49% complete
```

```

UUID : be623775:3e4ed7d6:c133873d:fbd771aa
Events : 0.5

```

Number	Major	Minor	RaidDevice	State	
3	8	8	0	spare rebuilding	/dev/sda8
1	8	6	1	active sync	/dev/sda6
2	8	7	2	active sync	/dev/sda7
4	8	5	-	faulty spare	/dev/sda5

- `mdadm --examine <partition>`: Displays detailed data about a component of a RAID array such as RAID level, total number of devices, number of working devices, and number of failed devices. For example, the output of `mdadm --examine /dev/sda6` shows the following:

```

/dev/sda6:
  Magic : a92b4efc
  Version : 00.90.00
  UUID : be623775:3e4ed7d6:c133873d:fbd771aa
  Creation Time : Mon Dec 18 07:39:05 2006
  Raid Level : raid1
  Device Size : 10241280 (9.77 GiB 10.49 GB)
  Array Size : 10241280 (9.77 GiB 10.49 GB)
  Raid Devices : 3
  Total Devices : 4
  Preferred Minor : 0

  Update Time : Mon Dec 18 07:40:01 2006
  State : active
  Active Devices : 2
  Working Devices : 3

```

```
Failed Devices : 0
Spare Devices : 1
Checksum : ee90b526 - correct
Events : 0.5
```

	Number	Major	Minor	RaidDevice	State	
this	1	8	6	1	active sync	/dev/sda6
0	0	0	0	0	removed	
1	1	8	6	1	active sync	/dev/sda6
2	2	8	7	2	active sync	/dev/sda7
3	3	8	8	3	spare	/dev/sda8

## Using MD Multipath

The hard drives in a system are connected to the rest of the system hardware via a disk controller. If the controller fails, the system can no longer communicate with the drives connected to it. However, some systems offer multipath disk access in which more than one controller is connected to the disks. If the active controller fails, a spare one replaces it, allowing continued access to the storage attached.

An example usage of MD Multipath is when a system is connected to a storage area network (SAN) via Fiber Channel Protocol or Cards. The multipath device can represent one interface that connects to the SAN using multiple physical cables. If one or more of the physical connections stops working or gets disconnected, the other physical cables are still active, and the storage is still accessible.

The Linux kernel offers Multiple Device (MD) Multipathing via its software RAID feature. MD Multipathing allows a device to be set up with multiple spares so that if the active device fails, I/O requests do not fail. If the active partition fails, the kernel activates one of the spare partitions as the active one.

To set up an MD Multipath device:

```
mdadm --create /dev/mdX --level=multipath --raid-devices=<num> <device list>
```

For example, use the following to set up `/dev/md0` with three drives, two of which become spares:

```
mdadm --create /dev/md0 --level=multipath --raid-devices=3 /dev/sda1 /dev/sdc1
↳ /dev/sdd1
```

The kernel monitors the failure of the partition and activates a spare when it fails. However, the `mdmpd` daemon from the `mdadm` RPM package must be running to automatically add a failed partition back to the array when it becomes available again.

## Understanding Clustering and GFS

In some enterprise infrastructures, high-performance, reliable, scalable servers and shared storage are necessary, with minimal downtime. Although RAID offers redundancy and NFS offers shared storage, they have limitations. For example, NFS transfer and access rates are slower than I/O to local disks and can have even slower rates depending on the number of simultaneous connections.

The Red Hat Cluster Suite offers application failover across multiple servers. Common servers that use clustering include web servers, database servers, and file servers such as GFS, or Global File Systems.

GFS is a scalable shared storage solution with I/O performance comparable to local disk access. It is usually combined with clustering to provide even more reliable storage with failover, redundancy, and simultaneous shared access to a GFS filesystem. When combined with clustering, the GFS filesystem is used on one or more file servers acting as the storage pool accessed by all the cluster nodes via a Storage Area Network (SAN). In addition to its ability to scale to meet the storage needs of hundreds or more servers simultaneously, the size of each GFS filesystem can be expanded while still in use.

The easiest way to start using the Red Hat Cluster Suite and Red Hat GFS is to install the packages from RHN using the Cluster Suite and GFS software channels. Refer to Chapter 3 for details on installing all the packages from a child software channel.

After installing the appropriate RPM packages, set up the cluster using the Cluster Configuration Tool (`system-config-cluster`) before configuring GFS. The exact configuration of Cluster Suite and GFS depends on a great deal of factors including the needs of your infrastructure, budget allocated to the system group, amount of shared storage needed plus extra for future expansion, and what type of application servers are to be run on the cluster servers. Refer to the *Documentation* and *Knowledgebase* sections of [redhat.com](http://redhat.com) for detailed instructions.

## Using Access Control Lists

On an ext3 filesystem, read, write, and execute permissions can be set for the owner of the file, the group associated with the file, and for everyone else who has access to the filesystem. These files are visible with the `ls -l` command. Refer to Chapter 4, “Understanding Linux Concepts,” for information on reading standard file permissions.

In most cases, these standard file permissions along with restricted access to mounting filesystems are all that an administrator needs to grant file privileges to users and to prevent unauthorized users from accessing important files. However, when these basic file permissions are not enough, *access control lists*, or *ACLs*, can be used on an ext3 filesystem.

ACLs expand the basic read, write, and execute permissions to more categories of users and groups. In addition to permissions for the owner and group for the file, ACLs allow for permissions to be set for any user, any user group, and the group of all users not in the group for the user. An effective rights mask, which is explained later, can also be set to restrict permissions.

To use ACLs on the filesystem, the `acl` package must be installed. If it is not already installed, install it via Red Hat Network as discussed in Chapter 3.

## Enabling ACLs

To use ACLs, they must be enabled when an `ext3` filesystem is mounted. This is most commonly enabled as an option in `/etc/fstab`. For example:

```
LABEL=/share /share ext3 acl 1 2
```

If the filesystem can be unmounted and remounted while the system is still running, modify `/etc/fstab` for the filesystem, unmount it, and remount it so the changes to `/etc/fstab` take effect. Otherwise, the system must be rebooted to enable ACLs on the desired filesystems.

If you are mounting the filesystem via the `mount` command instead, use the `-o acl` option when mounting:

```
mount -t ext3 -o acl <device> <partition>
```

## Setting and Modifying ACLs

There are four categories of ACLs per file: for an individual user, for a user group, via the effective rights mask, and for users not in the user group associated with the file. To view the existing ACLs for a file, execute the following:

```
getfacl <file>
```

If ACLs are enabled, the output should look similar to Listing 7.10.

### LISTING 7.10 Viewing ACLs

---

```
# file: testfile
# owner: tfox
# group: tfox
user::rwx
group::r-x
mask::rwx
other::r-x
```

---

To set or modify existing ACLs, use the following syntax:

```
setfacl -m <rules> <file>
```

Other useful options include `--test` to show the results of the command but not change the ACL and `-R` to apply the rules recursively.

Replace `<file>` with one or more space-separated file or directory names. Rules can be set for four different rule types. Replace `<rules>` with one or more of the following, and replace `<perms>` in these rules with one or more of `r`, `w`, and `x` (which stand for read, write, and execute):

- ▶ For an individual user:  
u:<uid>:<perms>
- ▶ For a specific user group:  
g:<gid>:<perms>
- ▶ For users not in the user group associated with the file:  
o:<perms>
- ▶ Via the effective rights mask:  
m:<perms>

The first three rule types (individual user, user group, or users not in the user group for the file) are pretty self-explanatory. They allow you to give read, write, or execute permissions to users in these three categories. A user or group ID may be used, or the actual username or group name.

#### CAUTION

If the actual username or group name is used to set an ACL, the UID or GID for it are still used to store the ACL. If the UID or GID for a user or group name changes, the ACLs are *not* changed to reflect the new UID or GID.

But, what is the *effective rights mask*? The effective rights mask restricts the ACL permission set allowed for users or groups other than the owner of the file. The standard file permissions are not affected by the mask, just the permissions granted by using ACLs. In other words, if the permission (read, write, or execute) is not in the effective rights mask, it appears in the ACLs retrieved with the `getfacl` command, but the permission is ignored. Listing 7.11 shows an example of this where the effective rights mask is set to read-only, meaning the read-write permissions for user `brent` and the group associated with the file are effectively read-only. Notice the comment to the right of the ACLs affected by the effective rights mask.

LISTING 7.11 Effective Rights Mask

```
# file: testfile
# owner: tammy
# group: tammy
user::rw-
user:brent:rw-           #effective:r--
group::rw-              #effective:r--
mask::r--
other::rw-
```

The effective rights mask must be set *after* the ACL rule types. When an ACL for an individual user (other than the owner of the file) or a user group is added, the effective rights

mask is automatically recalculated as the union of all the permissions for all users other than the owner and all groups including the group associated with the file. So, to make sure the effective rights mask is not modified after setting it, set it after all other ACL permissions.

If the ACL for one of these rule types already exists for the file or directory, the existing ACL for the rule type is replaced, not added to. For example, if user 605 already has read and execute permissions to the file, after the `u:605:w` rule is implemented, user 605 only has write permissions.

### Setting Default ACLs

Two types of ACLs can be used: *access ACLs*, and *default ACLs*. So far, this chapter has only discussed access ACLs. Access ACLs are set for individual files and directories. Directories, and directories only, can also have default ACLs, which are optional. If a directory has a default ACL set for it, any file or directory created in the directory with default ACLs will inherit the default ACLs. If a file is created, the access ACLs are set to what the default ACLs are for the parent directory. If a directory is created, the access ACLs are set to what the default ACLs are for the parent directory *and* the default ACLs for the new directory are set to the same default ACLs as the parent directory.

To set the ACL as a default ACL, prepend `d:` to the rule such as `d:g:500:rwX` to set a default ACL of read, write, and execute for user group 500. If any default ACL exists for the directory, the default ACLs must include a user, group, and other ACL at a minimum as shown in Listing 7.12.

#### LISTING 7.12 Default ACLs

---

```
# file: testdir
# owner: tfox
# group: tfox
user::rwx
group::r-x
mask::rwx
other::r-x
default:user::rwx
default:group::r-x
default:other::r--
```

---

If a default ACL is set for an individual user other than the file owner or for a user group other than the group associated with the file, a default effective rights mask must also exist. If one is not implicitly set, it is automatically calculated as with access ACLs. The same rules apply for the default ACL effective rights mask: It is recalculated after an ACL for any user other than the owner is set or if an ACL for any group including the group associated with the file is set, meaning it should be set last to ensure it is not changed after being set.

## Removing ACLs

The `setfacl -x <rules> <file>` command can be used to remove ACL permissions by ACL rule type. The `<rules>` for this command use the same syntax as the `setfacl -m <rules> <file>` command except that the `<perms>` field is omitted because all rules for the rule type are removed.

It is also possible to remove all ACLs for a file or directory with:

```
setfacl --remove-all <file>
```

To remove all default ACLs for a directory:

```
setfacl --remove-default <dir>
```

## Preserving ACLs

The NFS and Samba file sharing clients in Red Hat Enterprise Linux recognize and use any ACLs associated with the files shared on the server. If your NFS or Samba clients are not running Red Hat Enterprise Linux, be sure to ask the operating system vendor about ACL support or test your client configuration for support.

The `mv` command to move files preserves the ACLs associated with the file. If it can't for some reason, a warning is displayed. However, the `cp` command to copy files does *not* preserve ACLs.

The `tar` and `dump` commands also do *not* preserve the ACLs associated with files or directories and should not be used to back up or archive files with ACLs. To back up or archive files while preserving ACLs use the `star` utility. For example, if you are moving a large number of files with ACLs, create an archive of all the files using `star`, copy the `star` archive file to the new system or directory, and unarchive the files. Be sure to use `getfacl` to verify that the ACLs are still associated with the files. The `star` RPM package must be installed to use the utility. Refer to Chapter 3 for details on package installation via Red Hat Network. The `star` command is similar to `tar`. Refer to its man page with the `man star` command for details.

## Using Disk Quotas

Part of managing storage is determining how the available storage can be used. Although setting the size of filesystems such as `/tmp` and `/home` can limit storage for certain types of data, it is sometimes necessary to enable disk usage per user or per user group. This is possible with disk quotas. To use quotas, the `quota` RPM package must be installed. Refer to Chapter 3 for details on installing packages.

## Enabling Quotas

To use quotas, they must be enabled in `/etc/fstab`, which is read at boot time to mount filesystems. This enables quotas in the kernel booted for the system. To add as an option in `/etc/fstab`, for example (as the root user):

```
/dev/VolGroup00/LogVol01 /home          ext3      usrquota,grpquota      1 2
```

The `usrquota` mount option enables user quotas, and the `grpquota` option enables group quotas. One or both can be used. Either reboot the system to enable the quotas or remount each filesystem as root with the following command:

```
mount -o remount,acl,usrquota,grpquota,rw <mountpoint>
```

Once again, one or both of `usrquota` or `grpquota` can be used. In our example, `<mountpoint>` would be `/home`. To verify that the remount enabled quotas, execute the following command:

```
mount | grep <mountpoint>
```

or use such as `mount | grep home` if you are following the example. The output shows which mount options were used to mount the filesystem:

```
/dev/VolGroup00/LogVol01 on /home type ext3 (rw,acl,acl,usrquota,grpquota)
```

## Creating Quota Database Files

The first time the system is booted with quotas enabled in `/etc/fstab`, quotas are not turned on because the quota database files for the filesystem do not exist. The `quotacheck` command is used to create these files.

After rebooting with quotas enabled in `/etc/fstab` and before executing the `quotaon` command to turn on quotas, the filesystem must be initialized to use quotas. If they do not already exist, the `aquota.user` and `aquota.group` files are created in the root directory of the filesystem. These are database files used to enforce quotas.

Refer to the `quotacheck` man page for a list of all options and determine which options are best for your situation. By default, only user quotas are checked and initialized. If you need to initialize user group quotas as well, specify it with the `-g` option. A typical command to run with options, as the root user, would be:

```
quotacheck -uvg <devicename>
```

such as:

```
quotacheck -uvg /dev/VolGroup00/LogVol02
```

Because disk usage can change when the filesystem is mounted in read-write mode, it is recommended that `quotacheck` be run when the filesystem is mounted read-only. If the filesystem is mounted when `quotacheck` is run, `quotacheck` will try to mount it read-only before starting the scan. It then remounts it in read-write mode after the scan is complete.

If it is unable to mount it read-only, a message similar to the following appears:

```
quotacheck: Cannot remount filesystem mounted on /home read-only
so counted values might not be right.
Please stop all programs writing to filesystem or use -m flag to force checking.
```

If quotacheck can't remount the filesystem read-only before starting, you can force the quota check anyway by using the `-m` command-line option.

The quotacheck utility should be run on a regular basis to keep quotas accurate or after a system crash in which the filesystem was not unmounted cleanly. To make sure it is done on a schedule, setup a cron task that is run automatically at set times. Refer to Chapter 11, "Automating Tasks with Scripts," for details on setting up a cron task.

After creating the quota database files, be sure to turn quotas on as described in the next section. After the quota database files are created, subsequent boots with the `usrquota` and/or `grpquota` mount options in `/etc/fstab` will automatically have quotas turned on for those filesystems.

### Turning Quotas On and Off

Quotas can be turned on and off without rebooting the system with the `quotaon` and `quotaoff` commands, but only for filesystems that meet two conditions: The filesystem must be mounted at boot time with the `usrquota` and/or `grpquota` mount options in `/etc/fstab`, and the filesystem must have the `aquota.user` and/or `aquota.group` files in the root of the filesystem.

To turn quotas on for an already mounted filesystem, the `quotaon` utility can be used. As root, use the following to enable user and group quotas:

```
quotaon -vug <devicename>
```

To temporarily turn off quotas, execute the following command as root:

```
quotaoff -vug <devicename>
```

The `-vug` options specify that messages should be displayed showing that the quotas are being turned off as well as error messages if they exist and that both the user and group quotas should be turned off.

To verify that the quotas have been turned on or off, execute the `mount` command and read the mount options used such as the following: `/dev/VolGroup00/LogVol01 on /home type ext3 (rw,acl,acl,usrquota,grpquota)`

### Setting and Modifying Quotas

Quotas can be set per user, group, or filesystem with the `edquota` command. The user or group name can be used or the UID or GID for the user or group. To set or modify the quota for a user, execute the following as root:

```
edquota <username>
```

To set or modify the quota for a user group, execute the following as root:

```
edquota -g <groupname>
```

When the `edquota` command is executed, the default text editor is opened as determined by the `$EDITOR` environment variable. In Red Hat Enterprise Linux, the default editor is Vi. To set the default editor to a different editor, execute the following command, replacing `emacs` with the editor of your choice (this setting is per user):

```
export EDITOR="emacs"
```

When this command is executed, it only changes the default editor for that login session. When the system is rebooted, this setting is lost. To permanently change the default editor, add the command as a line to your `.bashrc` file in your home directory. The `.bashrc` file is only read when a user logs in, so to enable changes to the file after you have already logged in, execute the `source ~/.bashrc` command.

When setting quotas, there are two types of limits: *soft limits* and *hard limits*. When the soft limit is reached, the user is warned and allowed to exceed the soft limit for a *grace period*, which is set to 7 days by default in Red Hat Enterprise Linux. This grace period allows the user or group time to reduce disk usage and return to below the soft limit. A hard limit is the absolute maximum amount of disk usage the user or group is allowed. After it is reached, no more disk space is allocated to the user or group.

If a user or group still exceeds the soft limit after the grace period has expired, the soft limit is treated as a hard limit, and the user or group is not allowed additional disk usage until the disk usage falls below the soft limit.

When the `edquota` command is executed, the output looks similar to Listing 7.13, which shows content for modifying quotas for the user `tfox`.

#### LISTING 7.13 Setting Disk Quotas

---

```
Disk quotas for user tfox (uid 501):
  Filesystem                blocks      soft    hard  inodes  soft    hard
  /dev/mapper/VolGroup00-LogVol102 59403         0        0      0       0       0
```

---

There are seven columns of information. The first column shows the filesystem in question. The next three columns are for setting quotas according to block size, with the first being the current block usage for the user or group. The next two are for setting the soft and hard limits for block usage. The last three columns are for inode usage, with the first being the current usage for the user or group, and the last two being the soft and hard limits. Setting any of these limits to 0, the default, means there is no limit. The block and inode usage columns are for reference only and should not be modified. Change the values of the limits, save the file, and exit.

To modify the grace period for a filesystem, execute the following as root:

```
edquota -t
```

This grace period is used for all users and groups. To set the grace period for a specific user, execute the following as root, where <username> is a username or UID:

```
edquota -T <username>
```

To set the grace period for a specific user group, execute the following as root, where <groupname> is a group name or GID:

```
edquota -T -g <groupname>
```

## Displaying Quotas

To display all quotas along with user and group usage, execute the following as root:

```
repquota -a
```

The output should look similar to Listing 7.14.

LISTING 7.14 Reporting Disk Usage and Quotas

---

```
*** Report for user quotas on device /dev/mapper/VolGroup00-LogVol01
Block grace time: 7days; Inode grace time: 7days
```

User	used	Block limits			File limits			
		soft	hard	grace	used	soft	hard	grace
root	-- 189192	0	0		336	0	0	
bfox	-- 3216936	40000000	45000000		26383	0	0	
tfox	-- 36329868	40000000	45000000		56253	0	0	

---

## Summary

After reading this chapter, you now have an understanding of the many storage configuration schemes in Red Hat Enterprise Linux, some of which can be combined. Standard partitions are straightforward and necessary for some mount points but lack the option to resize without destroying the existing partitions. LVM is the default partitioning scheme for Red Hat Enterprise Linux. Logical volumes can be resized easily. Software RAID offers redundancy and some speed advantages. Global File Systems and clustering offer scalable, reliable storage for enterprises.