Getting Started

This tutorial assumes that the new physical hard drive has been installed on the system and is visible to the operating system. The best way to do this is to enter the system BIOS during the boot process and ensuring that the BIOS sees the disk drive. Sometimes the BIOS will provide a menu option to scan for new drives. If the BIOS does not see the disk drive double check the connectors and jumper settings (if any) on the drive.

Finding the New Hard Drive in CentOS 6

Assuming the drive is visible to the BIOS it should automatically be detected by the operating system. Typically, the disk drives in a system are assigned device names beginning hd or sd followed by a letter to indicate the device number. For example, the first device might be /dev/sda, the second /dev/sdb and so on.

The following is output from a system with only one physical disk drive:

```
# ls /dev/sd*  /dev/sda  /dev/sda1  /dev/sda2
```

This shows that the disk drive represented by /dev/sda is itself divided into 2 partitions, represented by /dev/sda1 and /dev/sda2.

The following output is from the same system after a second hard disk drive has been installed:

```
# ls /dev/sd*  /dev/sda /dev/sda1 /dev/sda2 /dev/sdb
```

As shown above, the new hard drive has been assigned to the device file /dev/sdb. Currently the drive has no partitions shown (because we have yet to create any). At this point we have a choice of creating partitions and file systems on the new drive and mounting them for access or adding the disk as a physical volume as part of a volume group. To perform the former continue with this chapter, otherwise read Adding a New Disk to a CentOS 6 Volume Group and Logical Volume for details on configuring Logical Volumes.

Creating Linux Partitions

The next step is to create one or more Linux partitions on the new disk drive. This is achieved using the fdisk utility which takes as a command-line argument the device to be partitioned:

su - # fdisk /dev/sdb Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel Building a new DOS disklabel with disk identifier 0xd1082b01. Changes will remain in memory only, until you decide to write them. After that, of course, the previous content won't be recoverable. Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite) WARNING: DOS-compatible mode is deprecated. It's strongly recommended to switch off the mode (command 'c') and change display units to sectors (command 'u'). Command (m for help):

As instructed, switch off DOS compatible mode and change the units to sectors by entering the c and u commands:

Command (m for help): c DOS Compatibility flag is not set Command (m for help): u Changing display/entry units to sectors

In order to view the current partitions on the disk enter the p command:

Command (m for help): p Disk /dev/sdb: 34.4 GB, 34359738368 bytes 255 heads, 63 sectors/track, 4177 cylinders Units = cylinders of 16065 * 512 = 8225280 bytes Sector size (logical/physical): 512 bytes / 512 bytes I/O size (minimum/optimal): 512 bytes / 512 bytes Disk identifier: 0xd1082b01 Device Boot Start End Blocks Id System

As we can see from the above fdisk output, the disk currently has no partitions because it is a previously unused disk. The next step is to create a new partition on the disk, a task which is performed by entering n (for new partition) and p (for primary partition):

Command (m for help): n Command action e extended p primary partition (1-4) p Partition number (1-4):

In this example we only plan to create one partition which will be partition 1. Next we need to specify where the partition will begin and end. Since this is the first partition we need it to start at the first available sector and since we want to use the entire disk we specify the last sector as the end. Note that if you wish to create multiple partitions you can specify the size of each partition by sectors, bytes, kilobytes or megabytes.

Partition number (1-4): 1 First sector (2048-67108863, default 2048): Using default value 2048 Last sector, +sectors or +size{K,M,G} (2048-67108863, default 67108863): Using default value 67108863

Now that we have specified the partition we need to write it to the disk using the w command:

Command (m for help): w The partition table has been altered! Calling ioctl() to re-read

partition table. Syncing disks.

If we now look at the devices again we will see that the new partition is visible as /dev/sdb1:

```
# ls /dev/sd*  /dev/sda /dev/sda1 /dev/sda2 /dev/sdb /dev/sdb1
```

The next step is to create a file system on our new partition.

Creating a File System on a CentOS 6 Disk Partition

We now have a new disk installed, it is visible to CentOS 6 and we have configured a Linux partition on the disk. The next step is to create a Linux file system on the partition so that the operating system can use it to store files and data. The easiest way to create a file system on a partition is to use the mkfs.ext4 utility which takes as arguments the label and the partition device:

```
# /sbin/mkfs.ext4 -L /backup /dev/sdb1  mke2fs 1.41.12 (17-May-2010)  Filesystem label=/backup  OS type: Linux  Block size=4096 (log=2)  Fragment size=4096 (log=2)  Stride=0 blocks, Stripe width=0 blocks  2097152 inodes, 8388352 blocks  419417 blocks (5.00%) reserved for the super user  First data block=0  Maximum filesystem blocks=4294967296  256 block groups  32768 blocks per group, 32768 fragments per group  8192 inodes per group  Superblock backups stored on blocks:  32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,  4096000, 7962624  Writing inode tables: done  Creating journal (32768 blocks): done  Writing superblocks and filesystem accounting information: done  This filesystem will be automatically checked every 36 mounts or  180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

Mounting a File System

Now that we have created a new file system on the Linux partition of our new disk drive we need to mount it so that it is accessible. In order to do this we need to create a mount point. A mount point is simply a directory or folder into which the file system will be mounted. For the purposes of this example we will create a /backup directory to match our file system label (although it is not necessary that these values match):

```
# mkdir /backup
```

The file system may then be manually mounted using the mount command:

```
# mount /dev/sdb1 /backup
```

Running the mount command with no arguments shows us all currently mounted file systems (including our new file system):

```
# mount  /dev/mapper/vg_CentOS6-lv_root on / type ext4 (rw)  proc on /proc type proc (rw)
```

sysfs on /sys type sysfs (rw) devpts on /dev/pts type devpts (rw,gid=5,mode=620) tmpfs on /dev/shm type tmpfs (rw,rootcontext="system_u:object_r:tmpfs_t:s0") /dev/sda1 on /boot type ext4 (rw) none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw) sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw) /dev/sr0 on /media/CentOS_6.0 x86_64 Disc 1 type iso9660 (ro,nosuid,nodev,uhelper=udisks,uid=500,gid=500, iocharset=utf8,mode=0400,dmode=0500) /dev/sdb1 on /backup type ext4 (rw)

Configuring CentOS 6 to Automatically Mount a File System

In order to set up the system so that the new file system is automatically mounted at boot time an entry needs to be added to the /etc/fstab file.

The following example shows an fstab file configured to automount our /backup partition:

```
  /dev/mapper/vg_centos6-lv_root /            ext4    defaults        1 1
UUID=0d06ebad-ea73-48ad-a50a-1b3b8ef24491 /boot  ext4    defaults        1 2
/dev/mapper/vg_centos6-lv_swap swap         swap    defaults        0 0  tmpfs
/dev/shm            tmpfs   defaults        0 0  devpts                  /dev/pts            devpts
gid=5,mode=620  0 0  sysfs                   /sys                sysfs   defaults        0 0  proc
/proc               proc    defaults        0 0  LABEL=/backup /backup      ext4    defaults        1 2
```