SysAdmin Notes for RHCSA

Somenath Sinha

August 2017

Contents

Part I

Installing RHEL Server

Chapter 1

Using Essential Tools

1.1 Man Command

man followed by *keyword* yields the manual page of that command.

```
$ man ls
```

man followed by option **-k** (for keyword) and then followed by a *keyword* yields a list of all the commands containing that keywork and a brief descripiton of that command.

```
$ man -k day
daylight (3)         - initialize time conversion information
dysize (3)           - get number of days for a given year
daylight (3p)        - set timezone conversion information
gettimeofday (2)     - get / set time
gettimeofday (3p)    - get the date and time
motd (5)             - message of the day
Net::Time (3pm)      - time and daytime network client interface
settimeofday (2)     - get / set time
Time::HiRes (3pm)    - High resolution alarm, sleep, gettimeofday, interval timers
```

The numbers next to the commands indicate which seciton of the man pages the command belongs to (based on their functionality). The actual section that the commands belong to can be determined by the use of

```
$ man man-pages
```

The relevant sections for SysAdmins are Section 1, 5 & 8. The sections are:

Section Number	Deals with	Description
1	Commands (Programs)	Those commands that can be executed by the user from within a shell.
2	System calls	Those functions which must be performed by the kernel.
3	Library calls	Most of the libc functions.
4	Special files (devices)	Files found in /dev.
5	File formats and conventions	The format for /etc/passwd and other human-readable files.
6	Games	
7	Overview, conventions, and miscellaneous	Overviews of various topics, conventions and protocols, character set standards, and miscellaneous other things.
8	System management commands	Commands like mount(8), many of which only root can execute.

To filter down the output of the **man -k** command, we can use **grep** to obtain only the relevant parts of the result on the basis of the appropriate section number in the man-pages.

This can be achieved using the pipe which feeds the output of the first command to the input of the second command.

```
$ man -k day | grep 3
daylight (3)         - initialize time conversion information
dysize (3)           - get number of days for a given year
daylight (3p)        - set timezone conversion information
gettimeofday (3p)    - get the date and time
Net::Time (3pm)      - time and daytime network client interface
Time::HiRes (3pm)    - High resolution alarm, sleep, gettimeofday, interval timers
```

1.2 Understanding Globbing and Wildcards

*****	-	Indicates any string.
?	-	Indicates any single character.
[···]	-	Indicates any character provided within brackets.
[!···]	-	Indicates any character *NOT* provided within brackets.
[a-f]	-	Indicates any character provided within the range of a to f.

1.3 Understanding Globbing and Wildcards

$ ls a*	-	Lists all files and folders (including contents of each folder) that start with "a"
$ ls *a*	-	Lists all files and folders that contain the string "a".
$ ls -d a*	-	Shows all files and folders that start with "a" but excludes the contents of each individual folder.
$ ls ??st*	-	Lists all files and folders that have "st" as the 3^{rd} and the 4^{th} character in their name.
$ [a-f]	-	Indicates any character provided within the range of a to f.

1.4 Understanding I/O Redirection and Pipes

1.4.1 I/O Redirection

File Descriptors:

STDIN	-	0	-	Standard Input	-	Represents the "file" for the Standard Input Device (generally Keyboard).
STDOUT	-	1	-	Standard Output	-	Represents the "file" for the Standard Output Device (generally the Monitor).
STDERR	-	2	-	Standard Error	-	Represents the "file" for the Standard Output Device (also, generally the Monitor).

Redirection:

STDIN	-	<	-	Feeds the file to the right of the "<" as input to the command on the left.
STDOUT	-	>	-	Stores the output of the command to the left of the ">" to the file indicated on the right. *OVERWRITES* the mentioned file.
STDOUT	-	>>	-	Stores the output of the command to the left of the ">>" to the file indicated on the right. *APPENDS* the mentioned file.
STDERR	-	2 >	-	Redirects the errors from the command mentioned on the left to the file on the right. *OVERWRITES* the mentioned file.

```
1 $ mail -s hi root < .
2 $ ls > myFile
3 $ ls -lh >> myFile
4 $ grep hi * 2> /dev/tty6
```

1 - *mail* is a simple command used to send messages. The command expects the message to terminate with a ".", so we feed it directly to the command, instead of providing any input.

4 - The STDERR is redirected to tty6 (a virtual terminal connected to the host). Can also be diverted to a file if needed, such as an errorLog.

1.4.2 Piping

The command `$ ps aux` shows us the overview of all the running processes on the host. However, it's too long to view all at once. In such situations, or wherever we need to feed the output of the first command to the input of the second command, we use the pipe operator. The command would then be `$ ps aux | less`.

The difference in the usage of the piping and redirection operators is that Pipe is used to pass output to another program or utility, while Redirect is used to pass output to either a file or stream.

1.5 Using I/O Redireciton and Piping

```
$ ps aux | awk '{print $2}'
$ ps aux | awk '{print $2}' | sort
$ ps aux | awk '{print $2}' | sort -n
```

The second column ($2) of the `$ ps aux` command contains the Process ID (PID), and if we only want to filter the output such that only the PID is shown, we simply use the **awk** filtering utility.

If we want to sort the output of the command, we use the **sort** utility, but it generally sorts as a string. To sort the output as a number, we use the option **sort -n**.

If you expect lots of errors for a particular command, but want to discard all errors and only see the output when successful, then simply redirect the STDERR to `/dev/null`, which is a special device that discards all data written to it, i.e., a dustbin for data.

```
$ find / -name "*.rpm"
$ find / -name "*.rpm" 2> /dev/null
```

The first command shows all output including errors, but the second command discards all errors and shows the rest.

```
$ some_command > /dev/null 2> &1
```

The above code redirects STDOUT to `/dev/null` thus destroying the output, and also redirects the STDERR (2>) to STDOUT (&1). Essentially, it discards all output - useful when we don't need the output but only need the command to execute.

```
$ ls / > file_list.txt
$ sort < file_list.txt > file_list_sorted.txt
```

The above command stores the contents of the root directory in *file_list.txt*. Then, the second command uses both input and output redirection! The input of the sort command is fed from *file_list.txt* and the corresponding output sent to *file_list_sorted.txt*.

Chapter 2

Essential File Management Tools

2.1 Understanding Linux File System Layout

root (/)	-	Contains all other directories.
/boot	-	Contains everything the system needs to start up
/usr	-	Contains program files
/etc	-	Contains configuration files
/home	-	Contains a user's files
/mnt	-	Used to manually mount devices
/media	-	Devices like optical discs get auto-mounted on the media directory

Unlike other OSs, the linux files system is designed as such that multiple devices can be mounted on the same file system hierarchy. Thus, it's possible to mount devices remotely as well!

2.2 Finding Files

The **find** command is used to find a file within a folder and its subdirectories. When the starting point of the search is the root directory (/) then find will search the entire file system. While the utility is extremely thorough, this may cause delays due to remote devices on the network mounted on the file system.

```
$ find / -name "passwd"
```

If you're trying to find the location of a binary file, a better command would be **which** command, as it directly shows the location of the binary, but be careful as it only works with binaries.

```
$ which passwd
/usr/bin/passwd
```

Contrastingly, the command **whereis** not only gives us the locaiton of the binary, but the location of the complete environment of the binary!

```
$ whereis passwd
passwd: /usr/bin/passwd /etc/passwd /usr/share/man/man1/passwd.1.gz
 ↪  /usr/share/man/man5/passwd.5.gz
```

Another similar utility is called **locate** which shows all files that have the string provided to it in its name. Note, however, that locate operates on a database, that must be updated (especially after the creation of a new file) to show relevant results.

```
# touch sinha
# ls
sinha
# locate sinha
/usr/share/vim/vim74/keymap/sinhala-phonetic_utf-8.vim
/usr/share/vim/vim74/keymap/sinhala.vim
# updatedb
# locate sinha
/home/somu/Documents/sinha
/usr/share/vim/vim74/keymap/sinhala-phonetic_utf-8.vim
/usr/share/vim/vim74/keymap/sinhala.vim
```

2.3 Understanding Links

inode - An inode is a datastructure that describes a file system object such as a file or a directory, containing both the disc block locations as well as the attributes of the file system object. The inodes are identified by their inode number.

Consequently, for us to access the files/directories, we need to be able to provide a name to the inodes, which are called hardlinks. A file may have more than one hardlink. Note that each hardlink is simply a different name provided tot he same inode. Ths, all hardlinks to the same file/directory have the same inode number. Hardlinks are one-directional only, i.e., the hardlink itself knows which inode it points to, but the inodes only know the total number of hardlinks that are associated with it, and not which exact ones are pointing to it. Since hardlinks point to some inode, they always need to stay on the same partition as the inode.

A symbolic link on the other hand, points to a hardlink instead of an inode. As such, it has a different inode number than the one that the hardlink points to. Thus, the hardlink and symbolic link can be on different partitions as well. It can even exist across servers. Whenever a hardlink is deleted, however, all the symbolic links pointing to it are rendered invalid.

2.4 Working with Links

The `ln` command is used to create both hardlinks and symbolic links. To create a symbolic link, we need only add the `-s` option. The `-i` option of the `ls` command shows us the inode number.

```
# ln /etc/hosts computers
# ls -il /etc/hosts computers
```

```
8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 computers
8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 /etc/hosts
# ln -s computers newcomputers
# ls -il /etc/hosts computers newcomputers
8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 computers
8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 /etc/hosts
27604468 lrwxrwxrwx. 1 root root   9 Sep  7 19:26 newcomputers -> computers
# rm -f computers
# ls -il /etc/hosts newcomputers
8388733 -rw-r--r--. 1 root root 158 Jun  7  2013 /etc/hosts
27604468 lrwxrwxrwx. 1 root root   9 Sep  7 19:26 newcomputers -> computers
# exit
exit
$ ln /etc/shadow mydata
ln: failed to create hard link ‘mydata’ => ‘/etc/shadow’: Operation not permitted
$ ls -l /etc/shadow
----------. 1 root root 1375 Sep  5 21:04 /etc/shadow
```

When the hardlink *computers* to the inode associated with `/etc/hosts` is deleted, the associated symbolic link of *newcomputers* becomes invalid.

Finally, RHEL 7 onwards, a user may only create a link to a file/directory that it at least has a read permission to. Thus, any user won't be able to create a link to `/etc/shadow` as it has no permissions for anybody.

2.5 Working with tar

tar stands for Tape Archive. The command is most commonly used to make backups of files by storing them in archives. Some of the options of `tar` are:

-c	-	create	-	typically has an extension of .tar
-t	-	show contents	-	show contents of the archive.
-x	-	extract		
-z	-	file	-	compress the archive using gzip. Typically has an extension of `.tgz`
-v	-	verbose	-	tell us what the utility is doing.
-f	-	file	-	option to indicate the name of the archive file.
-C	-	location	-	indicates where the archive is to be extracted.

```
$ tar -cvf /root/etc.tar /etc
```

The above command creates the `etc.tar` archive in the `/root` directory and puts the contents of `/etc` in that archive. Note that the file `etc.tar` has a `.tar` extension only because we provided it, and not because Linux mandates it (unlike windows). Thus, sometimes we may run across tar archives that don't have an extension and are hard to detect. So, in that case we use the file command, which tells us the type of a particular file.

```
$ file /root/etc.tar
/root/etc.tar: POSIX tar archive (GNU)
```

Note that the .tar archive only puts all the files of teh `/etc` directory in the file `tar.etc`, but doesn't actually compress anything. To enable compression the `-z` option of the `tar` command must be used.

```
$ tar -czf /root/etc2.tgz /etc
```

Before extracting the contents of a tar file, we might want to see its contents, which can be done using the `-t` option of the `tar` command. *NOTE: Some older versions of* `tar` *may require the -z option to enable working with gzip archives, even when simply using the archive and not creating it.*

```
$ tar -tvf /root/etc2.tgz
```

To actually extract the archive, we use `-x` option. To indicate the location where we want the extracted files to reside, we include the `-C` option. If this option is not present then the files will be extracted in the present directory.

```
$ tar -xvf /root/etc2.tgz -C /tmp
```

To extract only one file from the archive, we can simply provide the name of the file at the very end.

```
$ tar -xvf /root/etc2.tgz -C / etc/wgetrc
```

NOTE that in the above command, we use the relative path `etc/wgetrc` *because of the fact that the archive stores a relative file path for easy extraction in any folder.*

Chapter 3

Working with Text Files

3.1 Understanding Regular Expressions

<table>
<tr><th>Character</th><th>Definition</th><th>Example</th><th>Result</th></tr>
<tr><td>^</td><td>Start of a string</td><td>^abc</td><td>abc, abcdef, abc123</td></tr>
<tr><td>$</td><td>End of a string</td><td>abc$</td><td>abc, blahabc, 456abc</td></tr>
<tr><td>.</td><td>Any character except newline</td><td>a.c</td><td>abc, aac, a2c</td></tr>
<tr><td>|</td><td>Alteration</td><td>1|8</td><td>1,8</td></tr>
<tr><td>{...}</td><td>Explicit quantity of preceding character</td><td>ab{2}c</td><td>abbc</td></tr>
<tr><td>[...]</td><td>Explicit set of characters to match</td><td>a[bB]c</td><td>abc, aBc</td></tr>
<tr><td>(...)</td><td>Group of characters</td><td>(123){3}</td><td>123123123</td></tr>
<tr><td>*</td><td>Null or more of the preceding character</td><td>ab*c</td><td>ac, abc, abbbbbc</td></tr>
<tr><td>+</td><td>One or more of the preceding character</td><td>ab+c</td><td>abc, abbbbc</td></tr>
<tr><td>?</td><td>Null or one of the preceding character</td><td>ab?c</td><td>ac, abc</td></tr>
</table>

PEARSON IT CERTIFICATION livelessons

Figure 3.1: RegEx Cheat Sheet

3.2 Using common text tools

3.2.1 cat

The `cat` command prints the entire content of a file on to the terminal.

3.2.2 less

Sometimes the `cat` command is unsuitable, like in the case of extremely large files. In such cases, like the `/var/log/messages`, the default system log file, using `cat` won't work as the majority of the messages would scroll past fast. For such cases, `less` is a better utility. Search functionality is exactly the same as in the case of vim.

3.2.3 Head and Tail

Head

The `head` command by default shows us the first 10 lines of a text file. To see more or less lines, the `-n` option can be used.

```
$ head -n 20 file.txt
```

Tail

The `tail` command by default shows us the last 10 lines of a text file. To see more or less lines, the `-n` option can be used.

```
$ tail -n 5 file.txt
```

Combination of head and tail

The combination of these two commands can enable the viewing of text in between specific line numbers. The command below shows lines 16-20 of `file.txt`

```
$ head -n 20 file.txt | tail -n 5
```

3.2.4 cut

With the `cut` utility, we can print out a specific column form a text file. It assumes the columns are separated by Tabs. Which specific column is to be printed is set by using the `-f` option. For example, to only print the first column of a text file, we say:

```
$ cut -f 1 cities
```

To provide a different delimter, such as ":" we use the `-d` option followed by the delimiter of our choice.

```
$ cut -f 1 -d : /etc/passwd
```

3.2.5 sort

This command sorts the input provided in the order of the ASCII table. That means numbers first, captial letters next and finally the lower case letters.

```
$ cut -f 1 -d : /etc/passwd | sort
```

To sort on the basis of a specific criteria:

- **-n** - Sort on the basis of actual numberical value, instead of treating a number as a string.
- **-f** - Sort in a case insensitive manner.

3.2.6 tr

The tr command replaces certain characters with certain other characters. Thus, it's frequently used in conjunction with pipes to modify the output of a command.

```
$ echo hello | tr a-z A-Z
HELLO
$ echo hello | tr [:lower:] [:upper:]
HELLO
```

3.3 grep

grep is a filtering utility that only prints those lines that contain a certain expression matching the pattern provided by a *RegEx*.

```
$ ps aux | grep tracker
somu      10450  0.0  0.4 469796  9000 ?        SNl  10:06   0:00 /usr/libexec/tracker-miner-user-guides
somu      10465  0.0  0.6 536856 12012 ?        Sl   10:06   0:00 /usr/libexec/tracker-store
somu      10611  0.0  0.7 779816 13108 ?        SNl  10:06   0:00 /usr/libexec/tracker-extract
somu      10614  0.0  0.5 469800  9632 ?        SNl  10:06   0:00 /usr/libexec/tracker-miner-apps
somu      10615  0.0  0.7 710160 13204 ?        SNl  10:06   0:00 /usr/libexec/tracker-miner-fs
root      17396  0.0  0.0 112644   968 pts/0    R+   13:49   0:00 grep --color=auto tracker
```

Another use for grep is searching files. The syntax is grep <filename-pattern> <search-directory>.

To avoid errors notifying "is a directory", simply redirect errors to /dev/null.

```
$ grep lisa * 2> /dev/null
group:lisa:x:1001:
gshadow:lisa:!::
passwd:lisa:x:1001:1001::/home/lisa:/bin/bash
passwd-:lisa:x:1001:1001::/home/lisa:/bin/bash
```

```
services:na-localise     5062/tcp                # Localisation access
services:na-localise     5062/udp                # Localisation access
shadow:lisa:$6$0l/zSJkh$xjJNYNnj1rPs7Fq0hDWt8VucS0nLL82XrMYpmBnLF2DrzB2npFvCwxM9MJEHgCHCwvabCgEA17LK2aU0h9FIT/:1741
shadow-:lisa:password:17414:0:99999:7:::
```

3.3.1 wc

Counts the number of words, lines and characters.

-l	-	Counts the number of lines
-w	-	Counts the number of words
-m	-	Counts the number of characters
-c	-	Counts the number of bytes
-L	-	Counts the length of the longest line

To see the number of matched lines using `grep`, simply use:

```
$ ps aux | wc
188     2344    19581
```

3.3.2 grep -l

Grep by default returns the name of the matching file followed by the matching lines. This output can be made more readable by `grep -l` which lists all the files in the directory that matches the criteria.

```
$ grep lisa * 2> /dev/null
group:lisa:x:1001:
passwd:lisa:x:1001:1001::/home/lisa:/bin/bash
services:na-localise     5062/tcp                # Localisation access
services:na-localise     5062/udp                # Localisation access
$ grep -l lisa * 2> /dev/null
group
passwd
services
```

3.3.3 grep -i

The **-i** flag turns the grep command case-insensitive!

```
$ grep lisa * 2> /dev/null
group:lisa:x:1001:
passwd:lisa:x:1001:1001::/home/lisa:/bin/bash
services:na-localise     5062/tcp                # Localisation access
services:na-localise     5062/udp                # Localisation access
$ grep -i lisa * 2> /dev/null
group:lisa:x:1001:
```

```
passwd:lisa:x:1001:1001::/home/lisa:/bin/bash
services:ltctcp          3487/tcp                # LISA TCP Transfer Channel
services:ltcudp          3487/udp                # LISA UDP Transfer Channel
services:na-localise     5062/tcp                # Localisation access
services:na-localise     5062/udp                # Localisation access
```

3.3.4 grep -R

The usage of the `-R` flag puts grep in recursive mode, where the utility searches for the file in each subfolder as well.

```
$ grep -iR lisa * 2> /dev/null
group:lisa:x:1001:
lvm/lvm.conf:       # If using external locking (type 2) and initialisation fails, with
passwd:lisa:x:1001:1001::/home/lisa:/bin/bash
Binary file pki/ca-trust/extracted/java/cacerts matches
Binary file pki/java/cacerts matches
...
```

3.3.5 grep -v

Grep with a `-v` flag excludes the matching results. Here we can exclude the lines containing "Binary" using:

```
$ grep -iR lisa * 2> /dev/null | grep -v Binary
alternatives/jre_openjdk/lib/security/nss.cfg:handleStartupErrors =
    ignoreMultipleInitialisation
alternatives/jre_openjdk_exports/lib/security/nss.cfg:handleStartupErrors =
    ignoreMultipleInitialisation
brltty/fr-abrege.ctb:word civilisation        14-1236-16
brltty/fr-abrege.ctb:word civilisations        14-1236-16-234
brltty/fr-abrege.ctb:word généralisation        1245-1345-16
brltty/latex-access.ctb:  brailleTranslator.capitalisation = "6dot"
brltty/latex-access.ctb:
passwd:lisa:x:1001:1001::/home/lisa:/bin/bash
sane.d/canon_pp.conf:# Set a default initialisation mode for each port.  Valid modes are:
services:ltctcp          3487/tcp                # LISA TCP Transfer Channel
services:ltcudp          3487/udp                # LISA UDP Transfer Channel
...
```

3.4 sed and awk basics

3.4.1 sed

`sed` is an old utility that's used to process text. Many of it's functionalities can now be done using `grep` itself.

sed q

To see the first two lines of a file using `sed` we use:

```
$ sed 2q /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
```

sed -n

The `-n` flag makes `sed` print no output unless the `p` flag is also provided. Here, we use a Regular Expression *"^root"* to match only a certain part of the text and then the `p` flag to print only if that criteria is matched.

```
$ sed -n /^root/p /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

The above result could also be obtained with `grep "^root" /etc/passwd`.

Substitution with sed

Sed can be used to substitute text within a file using the `s` parameter. It's used as :

```
$ cat names
Somu
Arpi
Neha
Santy
Debu
$ sed -i 's/Santy/Dickwad/g' names
$ cat names
Somu
Arpi
Neha
Dickwad
Debu
```

The `-i` flag asks the modifications to be made in place. Otherwise the output (changed text) would've simply been displayed to the screen and then need to be redirected to a file for storage.

3.4.2 awk

awk is another utility that is especially useful when working with text files. It excels at operations like cutting out information.

Cutting out information using awk

For certain operations, the `awk` command is a lot more powerful than the `cut` utility. In the example below `cut` has a hard time recognizing the second field, while `awk` has no problem whatsoever!

```
$ ps aux | grep 'gdm'
root       1117  0.0  0.1 480248  4688 ?        Ssl  09:20   0:00 /usr/sbin/gdm
root       1333  0.8  1.2 328524 47220 tty1     Ssl+ 09:20   0:43 /usr/bin/X :0
 ↪  -background none -noreset -audit 4 -verbose -auth
 ↪  /run/gdm/auth-for-gdm-bgrBZH/database -seat seat0 -nolisten tcp vt1
root       1718  0.0  0.1 528944  5848 ?        Sl   09:20   0:00 gdm-session-worker
 ↪  [pam/gdm-password]
gdm        1737  0.0  0.1 458088  4152 ?        Sl   09:20   0:00 ibus-daemon --xim
 ↪  --panel disable
gdm        1741  0.0  0.1 373560  5424 ?        Sl   09:20   0:00 /usr/libexec/ibus-dconf
gdm        1745  0.0  0.2 438152  7772 ?        Sl   09:20   0:00 /usr/libexec/ibus-x11
 ↪  --kill-daemon
somu      12218  0.0  0.0 112664   972 pts/0    R+   10:47   0:00 grep --color=auto gdm
$ ps aux | grep 'gdm' | cut -f 2
root       1117  0.0  0.1 480248  4688 ?        Ssl  09:20   0:00 /usr/sbin/gdm
root       1333  0.8  1.2 328524 47220 tty1     Ssl+ 09:20   0:43 /usr/bin/X :0
 ↪  -background none -noreset -audit 4 -verbose -auth
 ↪  /run/gdm/auth-for-gdm-bgrBZH/database -seat seat0 -nolisten tcp vt1
root       1718  0.0  0.1 528944  5848 ?        Sl   09:20   0:00 gdm-session-worker
 ↪  [pam/gdm-password]
gdm        1737  0.0  0.1 458088  4152 ?        Sl   09:20   0:00 ibus-daemon --xim
 ↪  --panel disable
gdm        1741  0.0  0.1 373560  5424 ?        Sl   09:20   0:00 /usr/libexec/ibus-dconf
gdm        1745  0.0  0.2 438152  7772 ?        Sl   09:20   0:00 /usr/libexec/ibus-x11
 ↪  --kill-daemon
somu      12226  0.0  0.0 112664   968 pts/0    R+   10:48   0:00 grep --color=auto gdm
$ ps aux | grep 'gdm' | awk '{ print $2 }'
1117
1333
1718
1737
1741
1745
12248
```

Chapter 4

Connecting to a RHEL Server

4.1 Connecting to a Server with SSH

To connect to a server we use the `ssh` command. The Syntax is: `ssh <server-ip>`.

```
$ ssh 192.168.152.129
somu@192.168.152.129's password:
Last login: Mon Nov 13 12:37:26 2017 from 192.168.152.128
```

The default SSH port is **22**. To connect to SSH on a different port (common when server is exposed to the internet), is `ssh -p <port-number> <server-ip>`. Note that *if root login is disabled on the server,* we must also provide the username to login as. The syntax then becomes : `ssh -p <port-number> <username>@<server-ip>`.

```
$ ssh -p 2022 sander@ldap.rhatcertification.com
```

4.2 RSA Key fingerprint and known hosts

Upon each new connection the ssh daemon shows us the RSA key fingerprint of the host to verify if we're connecting to the right computer. If so, the host is added to a list of known hosts permanently, in `~/.ssh/known_hosts`.

When the key fingerprint of the server doesn't match the Key fingerprint on record, the system warns us from connecting! This may occur when the server has been reinstalled on the same IP address. Thus, the new key fingerprint won't match the old one. To fix this, simply remove the old entry from `~/.ssh/known_hosts`.

4.3 sshd_config

The details of the method of connection to a server is stored in the **sshd_config** file, located in `/etc/ssh/sshd_config`.

Some of the options are:

Option	Description	Default Value
Port	The port number which is used for SSH on that server	22
PermitRootLogin	Whether an user is allowed to login as *root* user via SSH	yes

If the root login on the server via SSH is disabled, it generally makes the server a little bit more secure!

4.4 Understanding SSH keys

To initiate the ssh connection, the **SSHD** service on the server is contacted by the client. In order to confirm it's identity, the server responds with it's own `/etc/ssh/ssh_host.pub` public key to the client. When the client's user has verified the key of the server, the public key fingerprint gets stored in the clients `~/.ssh/known_hosts` file. Finally, the user is asked for the password to log on to the server.

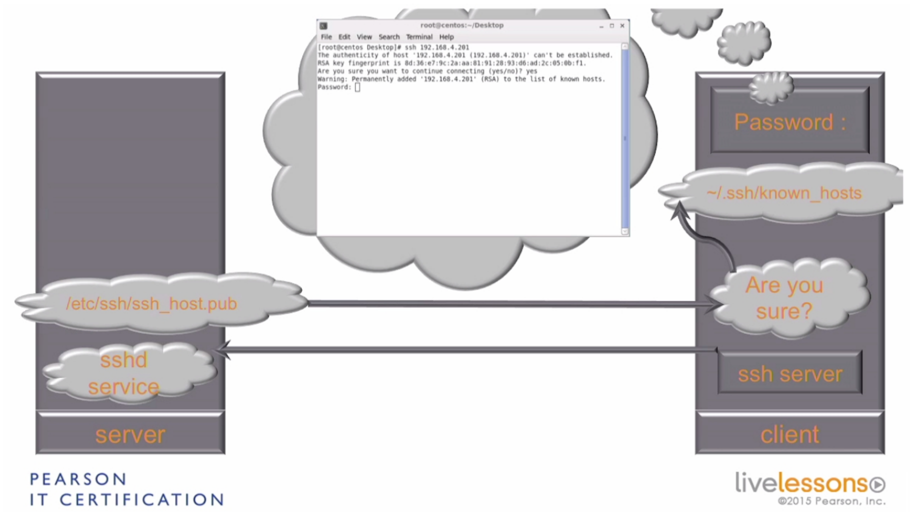

Figure 4.1: Server Authentication procedure

4.4.1 Client authentication without password

The client can also prove it's identity without a password by the use of a public key that it provides to the server. The private key of the user is stored in the home directory of the user `~/.ssh/id_rsa`. A packet encrypted with the private key is sent to the server which knows the user's public key. Some complex calculations based on this is performed on the authentication token sent from the client and if the identity is confirmed, then the user is logged in without needing a password.

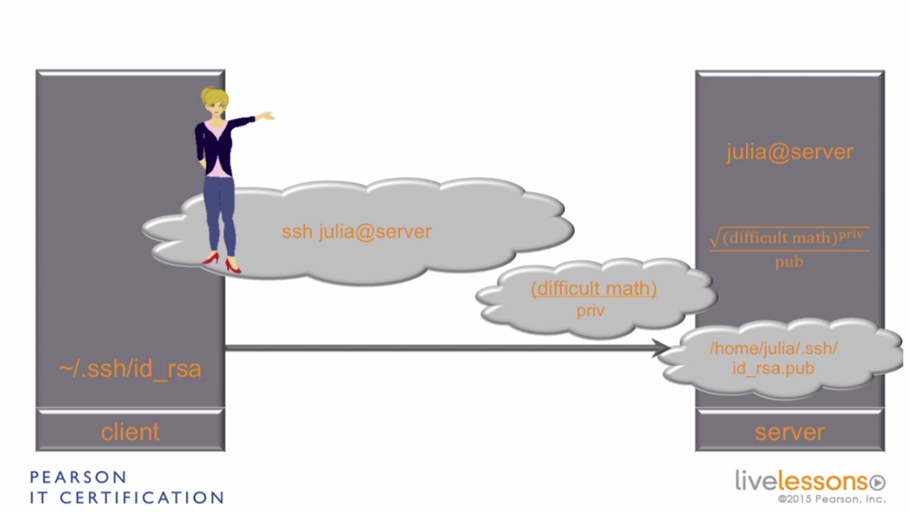

Figure 4.2: Public Key Client Authentication without password.

4.5 Using SSH Keys

SSH keys can be used to authenticate an user instead of a password. The private-public key pair can be generated using the `ssh-keygen` utility.

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/somu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/somu/.ssh/id_rsa.
Your public key has been saved in /home/somu/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:0C5uEHnAgJvzFEc0ulfL1YSygT/YF5UtL9lweTfPDAc somu@vmPrime.somusysadmin.com
The key's randomart image is:
+---[RSA 2048]----+
| ..==.    ooo .E. |
|. ..++o.oo+ + o.o|
| o.oo+++o..B . *o|
|+ ...===. o o   +|
| +. o +oS  .     |
|  .. o .         |
|      o          |
|     .           |
|                 |
+----[SHA256]-----+
```

4.5.1 Copying SSH keys

The SSH keys generated on the client now have to be copied to the server which requires authentication. This can be done using `ssh-copy-id`.

```
$ ssh-copy-id 192.168.152.129
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
somu@192.168.152.129's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '192.168.152.129'"
and check to make sure that only the key(s) you wanted were added.

$ ssh 192.168.152.129
Last login: Wed Nov 15 11:59:03 2017 from 192.168.152.128
```

While previous versions of `ssh-copy-id` didn't support specifying a port number, RHEL 7 onwards this feature is supported.

```
$ ssh-copy-id -p 2022 sander@ldap.rhatcertification.com
```

In case the public key is not recognized, it is possible to specify the public key using the `ssh-copy-id -i <publicKeyFile.pub>` flag.

4.5.2 Copying files to a server securely using SSH

Files can be copied to a server using SSH connection using the `scp` (secure copy) utility.

```
$ scp -P 22 names 192.168.152.129:~
names                  100%   28     8.1KB/s   00:00
```

NOTE that the directory on which the file has to be copied to on the server, (in this case the directory) has to be specified for the copy to be successful. Otherwise, `scp` just creates a local copy of the file with the name of the server as the filename.

Also, if the port has to be specified, the flag is `-P` which is in capital unlike `ssh` and `ssh-copy-id`.

Chapter 5

Managing Users and Groups

5.1 Understanding the need for Users

User accounts are not just to ensure that different people use resources with accountability and resource management. Several processes also have to execute with permissions given to them by their respective user accounts.

For example, the `apache` web server's processes and services execute under the permissions given to the `apache` account. This account doesn't have root privileges, which ensures that in case of security breaches of the apache user account, the culprit doesn't gain access to any critical resources that only an administrator or the root account should have access to.

5.2 User Properties

5.2.1 Username

A typical user info in the `/etc/passwd` file consists of the login information of several users, each with the following details :- `somu:x:1000:1000:Somu:/home/somu:/bin/bash`. Here, Somu is the username, the *x* in the second field references that a password has been stored for that username in the `/etc/shadow` file. The file contains the (one-way) encrypted password as well as several password related information such as password expiration dates, etc. Since the `/etc/shadow` file is only readable by the root user, it minimizes the security risk. Generally, only real user accounts need a password and system users (accounts used by processes to execute) don't.

/etc/shadow

While the `/etc/shadow` file contains the password of an user in an encrypted format, if the user account is new and doesn't yet have any password assigned to it, then the entry for it in /etc/shadow looks like:

```
$ cat /etc/shadow | grep lisa
lisa:!!:17485:0:99999:7:::
```

The second entry (!!) is where the encrypted password is usually stored. The double exclamation indicates that the *lisa* account hasn't set up a password yet.

5.2.2 UID

Each user on the system is setup with a unique UserID (UID). The root has a UID of 0, and normal users start with an UID of 1000 onwards. There are a total of 64,000 UIDs available for 2.4 kernels, and 4 billion for 2.6 kernels.

5.2.3 GID

On Linux, every user must be the member of at least one group, which is known as the **primary group** of the user, stored in the `/etc/passwd` file. On RHEL, that primary group has the same name as the username and the user is the only member of that group by default (i.e., private group). The list of Groups is stored in a file called `/etc/group`.

5.2.4 GECOS or comment field

This is a comment field that can contain anything the admin deems necessary. It generally contains information that makes the identification of each user easier.

5.2.5 Home Directory

The home directory refers to the location where the user is allowed to store files. For services, this folder is important because it defines the directory where the service can read and write files. While for regular users the home directory is typically inside `/home`, for services, they can be anywhere.

5.2.6 Default Shell

This is the shell (or command) that is executed on login of the user. The default value is `/bin/bash`.

5.3 Creating and Managing Users

5.3.1 Adding users

The default command for adding users on RHEL is `useradd`.

Option	Description
-e	Expiration date in the format YYYY-MM-DD. Sets the date on which the user account will be disabled.
-c	Comment that sets the contents of the GECOS field.
-s	Sets the default shell of the user. For example, a C programmer can use a shell such as TCSH.

```
$ sudo useradd -c "New Test User" -s /bin/tcsh -e 2017-12-31 laura
[sudo] password for somu:
$ # To verify the addition of the new user
$ tail -n 1 /etc/passwd
laura:x:1001:1001:New Test User:/home/laura:/bin/tcsh
```

id command

The `id` command prints the real and effective user information.

```
$ id
uid=1000(somu) gid=1000(somu) groups=1000(somu)
↪  context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

5.4 Understanding Group Membership

Groups are especially useful to enable users to share files with one another. These groups may be additional groups known as secondary groups.

The `/etc/passwd` file doesn't contain any reference to the secondary groups that the user is a part of, even though the `/etc/group` file lists that user as a member. Thus, the best way to obtain the groups the user is a part of is by using the `id` command.|

```
$ id lisa
uid=1002(lisa) gid=1002(lisa) groups=1002(lisa), 5009(sales)
```

5.5 Creating and Managing Groups

5.5.1 groupadd

The `groupadd` command is used to add a new group.

Option	Description
-g	Specify the GID of the group.

5.5.2 Adding users to a group

A user can be added to a group by directly editing the `/etc/group` file, or by| using the `moduser` command.

usermod

Option	Description
-g	Force assign the GID as the new default group of the user.
-G	Erase older list of supplementary groups and assign the given groups as the supplementary group of he user.
-a	Add the given group to the list of groups for the user.

Adding a user to a group using the `usermod` command is shown below.

```
$ sudo usermod -aG account laura
$ sudo usermod -aG 5010 lisa
$ tail -n 1 /etc/group
account:x:5010:laura,lisa
```

5.6 User and Group configuration files

Some of the important configuration files are:

Option	Description
/etc/passwd	Contains several details of the user, other than the password.
/etc/shadow	Contains the password hash and password properties for the user.
/etc/group	Contains the names of all the groups along with a list of all users in them.
/etc/login.defs	Contains the values (definitions) of several parameters used to create the user, such as password max days, min days, etc.
/etc/default/useradd	Contains the default values for several useradd parameters.
/etc/skel	When a user's home directory is created, the contents of `/etc/skel` is copied there, with the appropriate group of the user.

5.7 Managing Password properties

The user *root* can manage the password properties using two commands:

5.7.1 passwd

Option	Description
-d	Delete the current password.
-l	Lock the current password.
-u	Unlock the current password.
-e	Expire the current password - force user to change password during next login.
-x	Set the maximum lifetime of the password.
-n	Set the maximum lifetime of the password.
-w	Set days before expiration the user is warned.
-i	Set days after expiration the user account becomes inactive.

Locking and Unlocking passwords

```
$ sudo passwd -l laura
Locking password for user laura.
passwd: Success
$ su - laura
Password:
su: Authentication failure
$ sudo cat /etc/shadow | grep laura
laura:!!$6$0zDhsJet$q2...KRVKv8D2.:17486:0:99999:7::17531:
$ sudo passwd -u laura
Unlocking password for user laura.
passwd: Success
$ sudo cat /etc/shadow | grep laura
laura:$6$0zDhsJet$q2...KRVKv8D2.:17486:0:99999:7::17531:
$ su - laura
Password:
Last login: Thu Nov 16 13:40:45 IST 2017 on pts/0
$ whoami
laura
```

When an account is locked, the password hash for that user in the `/etc/shadow` file is prefixed with a `!!` to render it invalid and prevent authentication from succeeding (unless the root logs in as that user, which requires no password prompt).

5.7.2 chage

Option	Description
-l	List all password aging information.
-E	Set the account expiration date.
-m	Set the maximum lifetime of the password.
-M	Set the maximum lifetime of the password.
-W	Set days before expiration the user is warned.
-I	Set days after expiration the user account becomes inactive.

Setting the account expiration date

```
$ sudo chage -E 2017-12-31 laura
[sudo] password for somu:
[somu@cliServer ~]$ sudo cat /etc/shadow | grep laura
laura:$6$0zDhsJet$q2...KRVKv8D2.:17486:0:99999:7::17531:
$ sudo chage -l laura
Last password change                                : Nov 16, 2017
Password expires                                    : never
Password inactive                                   : never
Account expires                                     : Dec 31, 2017
Minimum number of days between password change      : 0
Maximum number of days between password change      : 99999
Number of days of warning before password expires : 7
```

The string *17531* represents the account expiration date in epoch time (seconds since Jan 1 1970).

Chapter 6

Connecting to a LDAP Server

6.1 Understanding LDAP

LDAP is an easy way to provide centralized authentication from a server. This way, many computers can be connected to a single LDAP server and the user accounts (and permissions) have to be set up only once!

LDAP stands for *Lightweight Directory Access Protocol*. It connects us to a hierarchical directory server. In the hierarchy (e.g., server.rhatcertification.com), there are top level domains such as *.com*, subdomain (rhatcertification) and leaf objects (lisa). Even though the structure is similar to DNS, the notation of LDAP is different. For every container object, we write `dc=<objectName>` (*dc* → Domain Component) and for leaf objects, it becomes `cn=<objectName>` (*cn* → Common Name). The complete format then becomes *cn=lisa,dc=rhatcertificaton,dc=com*.

An important part of connecting to an LDAP server is the **base context**. The base context, like the search domain of DNS, is the starting point where our client should look for objects. In this case, the base context is `dc=rhatcertification,dc=com`. Thus, for logging in to a server, the *cn*(lisa) is searched for within the base context.

6.1.1 /bin/login

The `login` service is used whenever the user requires authentication to connect to anything.

6.1.2 ldd

The `ldd` command (List Dynamic Dependencies) prints all the shared libraries required by a program.

```
$ ldd /bin/login
linux-vdso.so.1 =>  (0x00007ffc333e3000)
libpam.so.0 => /lib64/libpam.so.0 (0x00007f85cad8a000)
libpam_misc.so.0 => /lib64/libpam_misc.so.0 (0x00007f85cab86000)
libaudit.so.1 => /lib64/libaudit.so.1 (0x00007f85ca95d000)
libselinux.so.1 => /lib64/libselinux.so.1 (0x00007f85ca736000)
libc.so.6 => /lib64/libc.so.6 (0x00007f85ca373000)
```

```
libdl.so.2 => /lib64/libdl.so.2 (0x00007f85ca16e000)
libcap-ng.so.0 => /lib64/libcap-ng.so.0 (0x00007f85c9f68000)
libpcre.so.1 => /lib64/libpcre.so.1 (0x00007f85c9d06000)
/lib64/ld-linux-x86-64.so.2 (0x0000558e5f0bd000)
libpthread.so.0 => /lib64/libpthread.so.0 (0x00007f85c9ae9000)
```

The PAM library shown above (libpam) is akin to a plugin that adds additional functionality to the login utility (as well as several others). PAM stands for *Plugable Authentication Modules*. The configuration files for the authentication module is stored in /etc/pam.d directory. The /etc/pam.d/login is the configuration file for login utility.

6.1.3 PAM config file syntax

The PAM config files are each named after the services that require the usage of PAM. For example, the config file for the *login* service is called /etc/pam.d/login. Each file lists a bunch of rules in the syntax : <service-type> <control> <module-path> <arugments>.

Service Type

Service Type	Description
auth	Deals with user authentication via password (or other means like keys).
account	Non-authentication based account management.
password	Updating the authentication token of the user.
session	Modules listed here are used for setup/cleanup of a service for the user.

PAM Module Controls

Control	Description
requisite	Immediately causes failure when the module returns a status that isn't 'success'.
required	If the service returns a non-success status, then the operation fails ultimately, but only after the modules below it are invoked. This is to prevent a person with malicious intent from gaining knowledge of which module failed.
sufficient	If a *sufficient* module returns a 'success' status, the other modules below it that are also a part of 'sufficient' management group will not be invoked. In case of failure, another module listed 'sufficient' in the stack below it must succeed for the operation to succeed.
optional	Only causes failure if the rule stack contains only optional modules and all fail.
include	For the given service type, include all lines of that type from the provided configuration file.
substack	Same as *include* but when *done* and *die* actions are evaluated, they only cause skipping of the substack.

The login config file in /etc/pam.d contains the line:

```
auth       substack     system-auth
```

NOTE *: the entry for pam_ldap requires that the host should be able to use LDAP, which requires* pam_ldap *to be installed, and* authconfig-tui *to be executed.*

The system-auth file has rules for the common login procedure for any any process that deals with user authentication. This file in turn contains the lines :

```
auth        sufficient    pam_unix.so nullok try_first_pass
...
auth        sufficient    pam_ldap.so use_first_pass
```

The line `auth sufficient pam_unix.so` tells the system to look at the System login local authentication mechanism (`pam_unix.so`). If that is not successful, the system is instructed to use the LDAP PAM mechanism (in `pam_ldap.so`). Thus, the login process is contacting an LDAP server and trying to verify if the user account exists on that server.

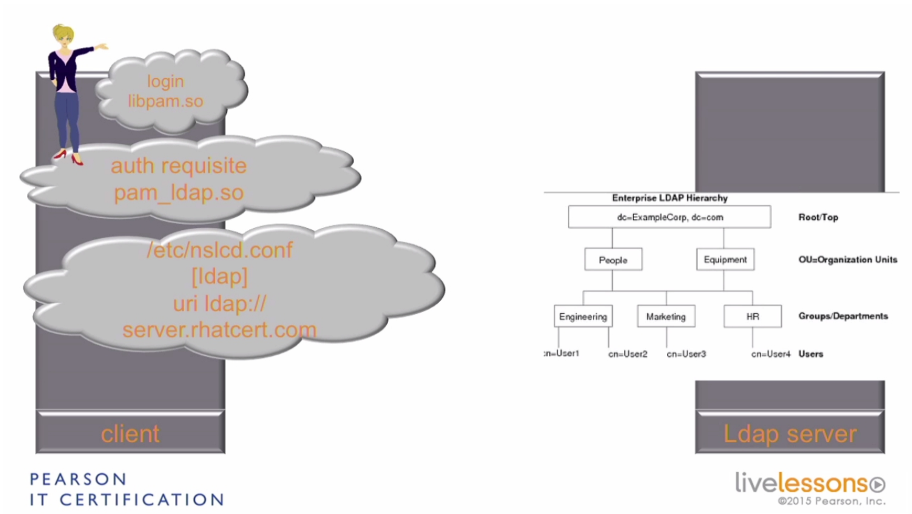

Figure 6.1: LDAP Authentication

Next, the LDAP configuration file (`/etc/nslcd.conf`) is read, which contains the URI of the LDAP Server (`ldap://server.rhatcert.com`). Finally, the client is able to connect to the LDAP server where there is a LDAP hierarchy that it can log into.

6.2 Telling your server where to find the LDAP Server

6.2.1 nscd

The Naming Service Cache Daemon needs to be installed to configure the connection of a server to an external LDAP server. It is the part of the OS that caches the information from external authentication mechanisms on the local machine.

6.2.2 nss-pam-ldapd

This sets up the local name resolution and local authentication and connects it to LDAP services.

6.2.3 pam_ldap

The libraries needed to make the local authentication aware of LDAP services.

6.2.4 authconfig-gtk

`authconfig` is an utility used to setup the server for external authentication. There are several variations of it, such as `authconfig`, `authconfig-tui` and `authconfig-gtk` (GUI based).

LDAP Search Base DN

The search base DN consists of Domain Components (dc) with commas as separators. Example : `dc=rhatcertification.com,dc=com`.

LDAP Server

The server needs to have a matching certificate to the one that the client receives on connection. This is only possible with a domain name, and not an IP address. The reason for this is the server name has to match the one in the certificate and the certificate can only have one name associated to it.

TLS Certificate

The use of a Transport Layer Security (TLS) certificate is important because unless it's used, the LDAP password is sent across the network unencrypted, which makes the entire system vulnerable. We also need to download the TLS certificate from the server (e.g., `ftp://server.rhatcertification.com/pub/slapd.pem`).

6.2.5 Switching to an LDAP user

The user can switch to an LDAP user just as easily as a local user using:

```
$ su - <ldap_username>
```

6.3 Understanding Automount

While it's possible to have the LDAP users use local directories on the server, generally an NFS hosts the home directories of these users. Thus, we have to automount the home directories of these users as if they're part of the local file system.

Let us consider a system where automounting is enabled, and the user wants to access a folder `/data/files`. If the folder `/data` is hosted on a remote file system and monitored by the automount process (called **autofs**), then there will have a file called `/etc/auto.master` containing the line:

```
/data        /etc/auto.data
```

The `/etc/auto.master` file only shows that the automount process recognizes the `/data` directory as an automount directory. This merely states that the mounting details for the data folder is present in its own file called `/etc/auto.data`. That file will contain:

```
files        -rw                 nfsServer:/data
```

The files directory is a subdirectory of the `/data` directory, and thus when the `/files` directory needs to be accessed, an NFS mounting operation needs to occur, with read write access on the nfsServer's (hostname) `/data` directory. Even though the user will be working on the NFS server, he/she will have no inkling of this happening behind the scene.

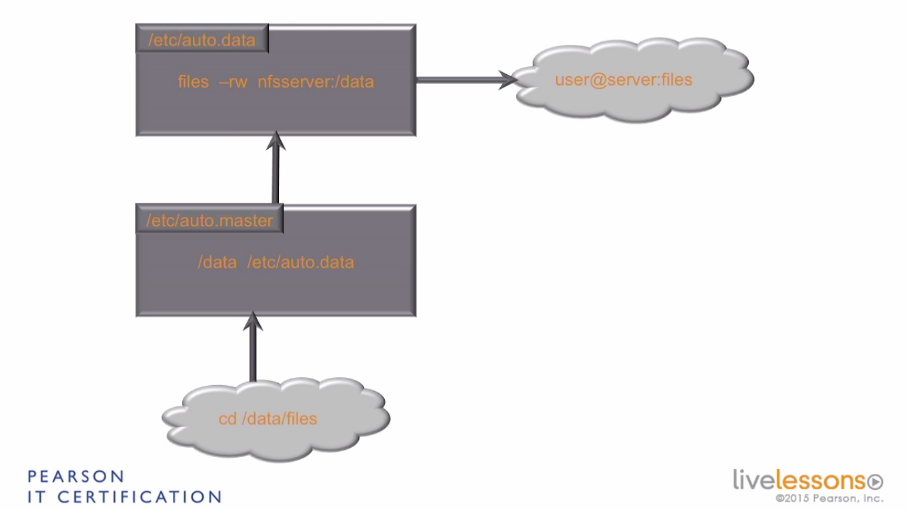

Figure 6.2: NFS Automount

6.3.1 Server selection for auto-mounting

Primarily two types of servers can accomplish the auto-mounting of home directories for LDAP users - NFS and Samba servers. In case of NFS server, the files will only be available on the local network. For access through the Internet, a Samba server has to be used.

6.3.2 Samba server's CIFS protocol to automount

Let us consider an LDAP user *ldapuser1* who has his home directory configured to `/home/guests/ldapuser1` in his user properties. When the user logs in, there will be a system call to go to the home directory for the users, which in turn calls `autofs` to mount the file system. It'll consult the `/etc/auto.master` file to find:

```
/home/guests        /etc/auto.guests
```

If anyone wants to visit that directory, the process should consult the `/etc/auto.guests` file, containing the mounting details with the UNC (Universal Naming Convention) path of the actual Samba server on the internet.

```
*         -fstype=cifs,username=ldapusers,password=password\
                 ://server.rhatcertification.com/data/&
```

So, if anyone goes to * (i.e., any directory in /data/guests), like /home/guests/ldapuser1 or /home/guests/ldapuser2 and so on, a CIFS (Common Internet File Sharing protocol, which uses Server Message Block [SMB, Used by Samba]) mount needs to occur, specified by fstype=cifs, with the given username and password. The address of the server is then provided.

What is of particular importance here is the matching of * and &. While the * wildcard selects whatever folder the user tried to enter, the & in the address is replaced with the corresponding text from user. Thus, if the user visits /home/guests/ldapuser1, the * is replaced with ldapuser1 and a matching folder is searched for on the server.

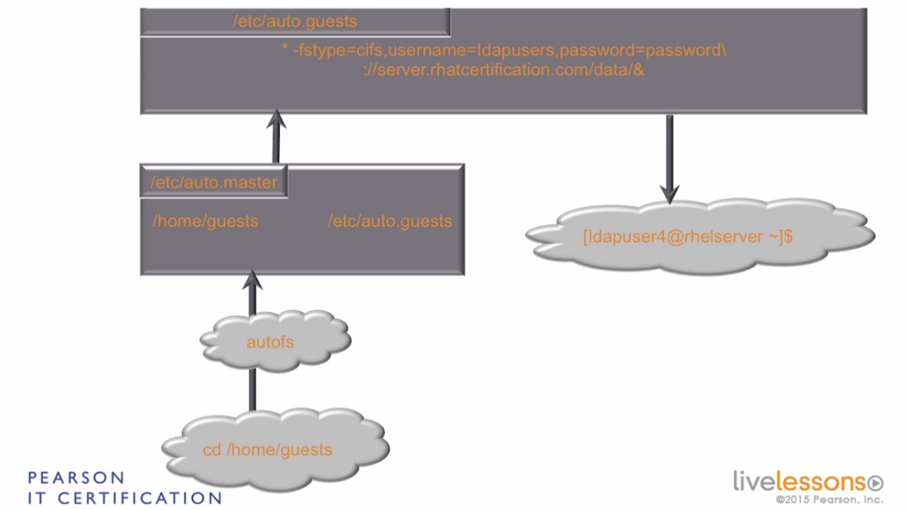

Figure 6.3: Samba Automount

6.4 Configuring Automount

To use automount, the *automount service* in the **autofs** package needs to be installed. The primary configuration file is /etc/auto.master. To automount the /etc/guests folder from a Samba server, we just need to specify in the file that:

```
/home/guests        /etc/auto.guests
```

Now we can add the mount options in its own individual file *auto.guests*, such that quick mounting and unmounting is possible.

Configuration on the Samba Server

The Samba server containing the data directory needs to have the following configuration in its /etc/samba/smb.conf :

```
[data]
comment = LDAP Users' home directories
path = /home/guests
public = yes
writable = no
```

6.4.1 NFS Server Automounting

In the case of a NFS mounted directory, the *auto.guests* file would look like:

```
*       -rw       nfsServer.domain.com:/home/guests/&
```

In case of either servers, the syntax remains the same. First we provide the name of the directory (`*`), then the mounting options (e.g., `-rw` in case of NFS) and finally the path to the real directory that has to be mounted on the local file system from that server.

6.5 Configuring NFS and Automount

6.5.1 yum search

The `yum` utility provides a searching function that searches the name, description, summary and url of all the packages available for a keyword.

```
# yum search nfs
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
* base: mirror.digistar.vn
* extras: mirror.dhakacom.com
* updates: mirror.digistar.vn
================================== N/S matched: nfs ===================================
libnfsidmap.i686 : NFSv4 User and Group ID Mapping Library
libnfsidmap.x86_64 : NFSv4 User and Group ID Mapping Library
libnfsidmap-devel.i686 : Development files for the libnfsidmap library
libnfsidmap-devel.x86_64 : Development files for the libnfsidmap library
nfs-utils.x86_64 : NFS utilities and supporting clients and daemons for the kernel NFS
 ↪  server
nfs4-acl-tools.x86_64 : The nfs4 ACL tools
nfsometer.noarch : NFS Performance Framework Tool
nfstest.noarch : NFS Testing Tool
```

6.5.2 Creating an NFS Server

The **nfs-utils** package is needed to setup an NFS server. The NFS configuration file is called `/etc/exports`. It specifies which file systems are exported to remote hosts (from the NFS server's perspective) and provides their respective mounting options. The contents of the *exports* file has to follow the syntax :

```
/data        *(rw,no_root_squash)
```

Here, `/data` is the name of the directory to be hosted on the NFS, with read/write permissions from all (*) IP addresses on the local network (since NFS only works on the local network). In cases it's not desirable to have the local machine's administrator act as the admin of the NFS server, then the way to perform this is called root squashing. In our case, we turn it off as we want the root user to retain administrative privileges on the NFS server as well.

6.5.3 Starting the NFS server

To start the service corresponding to the NFS server, we use the `systemctl` command.

```
$ systemctl start nfs
```

In case the service fails to start, the following command can provide hints about what went wrong :

```
# systemctl status -l nfs
nfs-server.service - NFS server and services
Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled; vendor preset:
↪  disabled)
Active: active (exited) since Tue 2017-11-21 10:00:04 IST; 24s ago
Process: 3178 ExecStart=/usr/sbin/rpc.nfsd $RPCNFSDARGS (code=exited, status=0/SUCCESS)
Process: 3173 ExecStartPre=/bin/sh -c /bin/kill -HUP `cat /run/gssproxy.pid`
↪  (code=exited, status=0/SUCCESS)
Process: 3172 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=1/FAILURE)
Main PID: 3178 (code=exited, status=0/SUCCESS)
CGroup: /system.slice/nfs-server.service

Nov 21 10:00:04 ldapserver.somuvmnet.local systemd[1]: Starting NFS server and
↪  services...
Nov 21 10:00:04 ldapserver.somuvmnet.local exportfs[3172]: exportfs: Failed to stat
↪  /data: No such file or directory
Nov 21 10:00:04 ldapserver.somuvmnet.local systemd[1]: Started NFS server and services.
```

Now we can see the mounting status of the NFS server hosted at localhost, using the `showmount -e localhost` command. Further, the NFS folder can be mounted manually using the `mount` command.

```
# showmount -e localhost
Export list for localhost:
/data *
# mount localhost:/data /mnt/nfs
# ls /nfs
localNFSfile1  localNFSfile2  localNFSfile3
```

6.5.4 Automounting NFS

For automounting NFS, we create a new entry in `auto.master` for a file `/etc/auto.nfs` :

```
/mnt/nfs        /etc/auto.nfs
```

This file contains the following mounting details:

```
files        -rw        localhost:/data/
```

Thus, when the user enters the *files* directory within the *nfs* directory, he'll find the same files as in the `/data` directory of localhost.

```
# cd nfs
# ls
# ls -lha
total 0
drwxr-xr-x. 2 root root  0 Nov 21 11:59 .
drwxr-xr-x. 3 root root 17 Nov 21 11:59 ..
# cd files
# ls
nfs1  nfs2  test1
# cd ..
# ls
files
```

Note that the `/nfs/files` directory isn't actually created before the user tries to enter the *files* directory.

6.6 Modifying nslcd Configuration

6.6.1 Naming Services LDAP Client Daemon

The `nslcd` is a service that connects the local file system to the external LDAP server. The status of the `nslcd` can be checked using:

```
$ systemctl status nslcd
nslcd.service - Naming services LDAP client daemon.
Loaded: loaded (/usr/lib/systemd/system/nslcd.service; enabled; vendor preset: disabled)
Active: active (running) since Mon 2017-11-20 12:03:06 IST; 5h 59min ago
Process: 1108 ExecStart=/usr/sbin/nslcd (code=exited, status=0/SUCCESS)
Main PID: 1151 (nslcd)
CGroup: /system.slice/nslcd.service
        1151 /usr/sbin/nslcd
```

/etc/nsswitch.conf

For every LDAP user, their identity needs to be known to the local system. This is based on the configuration stored in `/etc/nsswitch.conf`. In this file, there is a line:

```
passwd:        files        sss        ldap
```

This represents the order in which the sources of user account are searched for user related information. `sss` is an older service no longer used by RHEL-7. Finally, it looks for the information in LDAP using `nslcd`.

PAM using nslcd

PAM is responsible for the actual authentication system that ensures the LDAP server is known to the authentication mechanism. This entire process is achieved using *nslcd*.

6.6.2 /etc/nslcd.conf

This file contains the information stored by using the `authconfig-gtk` command and has options to configure the `nslcd` such that the LDAP server can be connected to and used.

Option	Description	Example Value
uri	The Uniform Resource Identifier of the LDAP Server.	ldap://server.rhatcertification.com
base	The base context of the LDAP Server	dc=rhatcertification,dc=com
ssl	Whether to use SSL/TLS. If start_tls is given, via the use of StartTLS, an insecure connection is upgraded to a secure one.	start_tls
tls_cacertdir	Location of the downloaded certificate of the LDAP Server.	`/etc/openldap/cacerts`

In case of any problems with using LDAP, the `/var/log/messages` file may contain hints that may indicate what's wrong.

Chapter 7

Managing Permissions

7.1 Understanding Ownership: Users, Groups and Others

The permissions for any file/folder in Linux can be viewed by using `ls -l` :

```
$ cd /home
$ ls -l
total 4
drwx------.  3 lisa lisa    78 Nov 15 21:32 lisa
drwx------.  3 1002 sales   78 Nov 15 21:36 rogue
drwx------. 19 somu somu  4096 Nov 20 19:33 somu
drwx------.  5 2002   101  128 Nov 19 23:36 testUsr
```

The format of the output is :
`<Permissions> <link-count of a file/no of files in directory> <owner> <group-owner> <file-size> <date & time of last modification> <file-name>`

7.1.1 Permissions

The first character in the permissions section, is the file type. The following file types are the most common:

Notation	Description
d	A directory
-	A regular file
l	A symlink/softlink

The rest of the permissions section is divided into three parts: the user's permissions, the group's permissions and other's permissions. The first 3 characters after the first one represents the user's permissions, the next 3 the group's and the final the other's. The possible values of these are:

Notation	Description
r	Read the file/directory
w	Write to the file/directory
x	Execute the file/Access to the directory
-	Permission NOT granted

7.1.2 Ownership

In linux, every file and directory (which is a *special* kind of file has an owner, as well as an associated group-owner. The owner is the user who created the file (unless specifically changed). The filesystem defines the permission set for the **owner**, the associated **group** and the rest of the users, called **others**.

While determining what set of permissions a user has to a file, linux first checks if the user is the owner. If so, the associated permissions are applied. If not, linux checks to see if the user belongs to the group which owns the file. If so, the group permissions on the file are granted. If both of these fail, then the user is determined to be '*other*' and the appropriate permissions applied. Of course, this requires the algorithm to be *exit-on-match*.

7.2 Changing file ownership

Let us consider a directory `/data` with the following structure:

```
$ ls -l
total 0
drwxr-xr-x. 2 root root 6 Nov 21 14:50 accounts
drwxr-xr-x. 2 root root 6 Nov 21 14:50 sales
```

The user 'root' has `rwx` permissions (all), while the group 'root' as well as others have only 'rw' (read/execute) permissions. None of them can write to the files in either of these directories by default.

7.2.1 chgrp

Now, it's reasonable to assume that everyone in sales should have write access to the sales directory, while everyone in accounts department should have write access to the group directory. Thus, we set these permissions using the `chgrp` command and setting the appropriate groups as the group-owner of these directories.

```
# ls -l
total 0
drwxr-xr-x. 2 root root 6 Nov 21 14:50 account
drwxr-xr-x. 2 root root 6 Nov 21 14:50 sales
# chgrp sales sales
# chgrp account account
# ls -l
total 0
drwxr-xr-x. 2 root account 6 Nov 21 14:50 account
drwxr-xr-x. 2 root sales   6 Nov 21 14:50 sales
```

The syntax for `chgrp` is `chgrp <group> <file/directory>`.

7.2.2 chown

The HoDs of these individual groups should be assigned as the owners of these directories. To assign them as such, we use the `chown` command.

```
# chown lori account
# chown lisa sales
# ls -l
total 0
drwxr-xr-x. 2 lori account 6 Nov 21 14:50 account
drwxr-xr-x. 2 lisa sales   6 Nov 21 14:50 sales
```

The syntax for the chown command is : chown <user> <file/directory>.
To change both the user and the group at once, the syntax becomes :
chown <user>:<group> <file/directory>.

7.3 Understanding Basic Permissions

Permission	**Files**	**Directories**
r	Opening and outputting a file.	List files in a directory. The user **can't** read all files in that directory. For that, he needs read access on the individual files.
w	Modify contents of the file	Modify contents of the directory, i.e., add, delete, move, etc. files in that directory.
x	If the contents of the file is executable, the user can execute it.	User can cd into the directory.

The fact that no file on a linux system has an executable permission by default is one of the core factors that makes th OS so secure. For example, even if a user were to get an email attachment with malware, it won't be able to run without execute permissions!

7.4 Managing Basic Permissions

7.4.1 chmod

The chmod command is used to change the permissions for a file/directory in linux. The user is represented by the letter *u*, the group by the letter *g* and others by *o*. The permissions themselves are represented by:

Permission		**Value**
r	=	4
w	=	2
x	=	1

In *absolute mode*, the individual permissions are added for each category of owner (r/g/o) and then provided to the chmod command to alter the permissions. Each category receives a value from the following table, representing a set of permissions.

Value	Permissions		*Breakdown*
7	Read, Write & Execute	`rwx`	(4+2+1)
6	Read & Write	`rw-`	(4+2)
5	Read & Execute	`r-x`	(4+1)
4	Read only	`r--`	(4)
3	Write & Execute	`-wx`	(2+1)
2	Write only	`-w-`	(2)
1	Execute only	`--x`	(1)
0	None	`---`	(0)

So, the syntax of `chmod` becomes: `chmod <val> <filename>`. An alternative method of applying permissions (called *relative mode*) is directly adding or subtracting permissions in the format:

```
chmod u<+-><rwx>,g<+-><rwx>,o<+-><rwx> <file-name>
```

```
$ chmod 750 myFile
$ chmod u+x,g-r,o-wx myFile2
$ chmod 0-x myFile3
```

Now, in our example, we want the HoD to have all permissions, the group to have rw permissions and others to have no access. Then we can set it using:

```
# ls -l
total 0
drwxr-xr-x. 2 lori account 6 Nov 21 14:50 account
drwxr-xr-x. 2 lisa sales   6 Nov 21 14:50 sales
# chmod 760 account
# chmod g+w-x,o-rx sales
# ls -l
total 0
drwxrw----. 2 lori account 6 Nov 21 14:50 account
drwxrw----. 2 lisa sales   6 Nov 21 14:50 sales
```

The permissions can also be set at once using `chmod 760 account sales`.

7.5 Understanding Special Permissions

Permission	*Symbol*	*Value*	**Files**	**Directories**
Set User ID	`u+s`	4	Run executable file as Owner	—
Set Group ID	`g+s`	2	Run executable file with permissions of Group-Owner	Inherit group ownership to all newly created items in the folder.
Sticky Bit	`+t`	1	—	Allows to delete files in the directory only if user is the owner or parent-directory-owner (or root).

SetUID : This is a special case where we grant the file special permission to be executed by any group or others (that have execution permission on the file) as if the owner of the file were running it. So, *the file executes with the same permission set as that of the owner*.

SetGID : This is a special case where we grant the file special permission to be executed by any user or others (that have execution permission on the file) as if the group-member of the file were running it. So, *the file executes with the same permission set as that of the group*.

Both SetUID and SetGID are dangerous permissions when applied to file and should be avoided if possible!

Sticky Bit : While it has no effect when applied on a file, when applied to a directory, especially in case of shared directories, one user cannot delete the file of another user (owner of the file), unless the user is owner of the directory or root.

7.6 Managing Special Permissions

Let us consider a shell script resides in the home directory of user *lisa* that deletes everything on the system:

```
#!/bin/bash
echo "Hi, do you wanna play a game?!"
read

rm -rf /
```

Generally, whenever a non-admin is going to execute this script, the only thing that'll be deleted would be user files (in directories the user has write access to), specifically the user home directory and the shared directories where the user has write access.

```
# chmod u+s game
# ls -l | grep game
-rwsr--r--. 1 root root 77 Nov 22 19:48 game
```

However, if the file were to be executed with the UID of an admin user, with root access, the `rm -rf /` command would cause critical damage. This is why both SetUID and SetGID are so dangerous!

7.6.1 Finding a file with a particular set of permissions

The find command is capable of finding a bunch of files where the permission set matches a format. We do this by:

```
# find / -perm /4000
find: '/proc/2998/task/2998/fd/6': No such file or directory
find: '/proc/2998/task/2998/fdinfo/6': No such file or directory
find: '/proc/2998/fd/6': No such file or directory
find: '/proc/2998/fdinfo/6': No such file or directory
/usr/bin/fusermount
/usr/bin/su
/usr/bin/umount
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/staprun
```

```
/usr/bin/mount
/usr/bin/pkexec
/usr/bin/crontab
/usr/bin/passwd
/usr/sbin/pam_timestamp_check
/usr/sbin/unix_chkpwd
/usr/sbin/usernetctl
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/lib64/dbus-1/dbus-daemon-launch-helper
/usr/libexec/abrt-action-install-debuginfo-to-abrt-cache
/home/lisa/game
```

Only special files are given this privilege, such as the `/usr/bin/passwd` binary executable. This is the files that enables us to change the password for a user. Now, to accomplish this the password has to be stored in an encrypted form in the `/etc/shadow` file with the following permissions:

```
# ls -l /etc/shadow
----------. 1 root root 1122 Nov 25 16:55 /etc/shadow
```

Thus, the *passwd* binary needs the root user privileges to make the `/etc/shadow` file temporarily editable by itself.

7.6.2 Setting Group ID for a directory

Let us consider the following scenario. User lisa is a member of the *account* group and the folder */data* has the following permissions:

```
#ls -l
total 0
drwxrwx---. 2 lori account 6 Nov 25 17:35 account
drwxrwx---. 2 lisa sales   6 Nov 25 17:26 sales
# su - lisa
Last login: Sat Nov 25 17:31:57 IST 2017 on pts/0
$ cd /data/account/
$ touch lisa1
$ ls -l
total 0
-rw-rw-r--. 1 lisa lisa 0 Nov 25 17:35 lisa1
```

The file */data/account/lisa1* has it's group owner set to the personal group of lisa. This means that the other members of the group *account* don't have write permission to that file. This is not acceptable in a shared group folder where multiple users have to edit the same file.

```
$ su - laura
Password:
Last login: Thu Nov 16 13:42:44 IST 2017 on pts/0
$ cd /data/account
$ echo "Added a line" >> lisa1
-bash: lisa1: Permission denied
```

This is why **Set group id** for a folder is so useful - so that each file created by the user in that directory, is by default editable by all the users in that group!

```
# ls -l
total 0
drwxrwx---. 2 lori account 19 Nov 25 17:35 account
drwxrwx---. 2 lisa sales    6 Nov 25 17:26 sales
# chmod g+s account
# ls -l
total 0
drwxrws---. 2 lori account 19 Nov 25 17:35 account
drwxrwx---. 2 lisa sales    6 Nov 25 17:26 sales
# su - lisa
Last login: Sat Nov 25 17:35:39 IST 2017 on pts/0
$ cd /data/account
$ touch lisa2
$ ls -l
total 0
-rw-rw-r--. 1 lisa lisa    0 Nov 25 17:35 lisa1
-rw-rw-r--. 1 lisa account 0 Nov 25 17:45 lisa2
$ echo "line added by lisa" >> lisa2
$ su - laura
Password:
Last login: Sat Nov 25 17:41:55 IST 2017 on pts/0
$ cd /data/account
$ echo "line added by laura" >> lisa2
$ cat lisa2
line added by lisa
line added by laura
```

7.6.3 Sticky Bit

When the sticky bit has been set the user can only delete a file if he/she's the owner of the file or the owner of the directory. This makes it invaluable in cases of shared directories, where each user needs write access to all files, and thus automatically gets the permission to delete any file he can write to!

In the case of the *account* directory, the owner of the file `lisa1` is lisa. Thus, the user laura can't delete it.

```
# ls -l
total 0
drwxrws---. 2 lori account 32 Nov 25 17:45 account
drwxrwx---. 2 lisa sales    6 Nov 25 17:26 sales
# ls -l account
total 4
-rw-rw-r--. 1 lisa lisa     0 Nov 25 17:35 lisa1
-rw-rw-r--. 1 lisa account 39 Nov 25 17:46 lisa2
# chmod +t account
# ls -l
total 0
drwxrws--T. 2 lori account 32 Nov 25 17:45 account
drwxrwx---. 2 lisa sales    6 Nov 25 17:26 sales
# su - laura
Last login: Sat Nov 25 17:53:25 IST 2017 on pts/0
$ cd /data/account
$ ls -l
total 4
-rw-rw-r--. 1 lisa lisa     0 Nov 25 17:35 lisa1
-rw-rw-r--. 1 lisa account 39 Nov 25 17:46 lisa2
```

```
$ rm -f lisa1
rm: cannot remove ‘lisa1’: Operation not permitted
$ su - lori
Password:
$ cd /data/account
$ rm -f lisa1
$ ls -l
total 4
-rw-rw-r--. 1 lisa account 39 Nov 25 17:46 lisa2
```

However, the user laura is able to delete it as she's the owner of the (parent) folder *account*.

7.6.4 Lowercase 's' or 't' vs Uppercase in permissions

The uppercase in case of *Set UserID/ Set GroupID/ Sticky Bit* indicates that that particular user/group or others don't have execute permissions on that directory. If however, they do have execute permissions then the 'S'/'T' is converted to lowercase, to indicate that there is an 'x' hidden behind the 's' or 't'.

```
# mkdir test
# ls -l
total 0
drwxrws--T. 2 lori account 19 Nov 25 17:57 account
drwxrws--T. 2 lisa sales    6 Nov 25 17:26 sales
drwxr-xr-x. 2 root root     6 Nov 25 18:15 test
# chmod 3770 *
# ls -l
total 0
drwxrws--T. 2 lori account 19 Nov 25 17:57 account
drwxrws--T. 2 lisa sales    6 Nov 25 17:26 sales
drwxrws--T. 2 root root     6 Nov 25 18:15 test
# chmod o+x,g-x test
# ls -l
total 0
drwxrws--T. 2 lori account 19 Nov 25 17:57 account
drwxrws--T. 2 lisa sales    6 Nov 25 17:26 sales
drwxrwS--t. 2 root root     6 Nov 25 18:15 test
```

An example of a folder with sticky bit set by default is `/tmp` where all users must be allowed to write files, but we don't want users to delete the files of other users.

7.7 Understanding ACLs

Access Control Lists are a way to permit allocation of permissions to a file/directory to more than one user or group. Normally, a file has only one user who is owner and only one group with a certain permission set. With ACLs it's possible to set different set of permissions to different groups/users! They can also be used to setup the default permissions for all newly created files/directories for any directory.

7.7.1 Mount options

To actually user ACLs, the **acl mount** options must be set. This can be done using either of `/etc/fstab` or **systemd**.

tune2fs for Ext file systems

tune2fs is an utility that lets us set adjustable file system parameters for the default Ext file system of RHEL/CentOS 7. This makes it possible to put the mount options *not* in a seperate file, but make it a property of the file system itself. Thus, if the file system is ever migrated to another server, the properties will be moved with it and not need to be set up again!

XFS

In XFS, there is no need for mounting options as it's a default mount option.

7.7.2 Commands

There are two primary commands to use ACLs: **setfacl** - (Set File Access Control Lists) and **getfacl** - (Get File Access Control Lists) are the two commands used to work with ACLs.

```
$ setfacl -m g:sales:rx /data/account
```

The critical part of this command is the part `g:sales:rx` which tells us that the group *sales* is getting the read and execute permissions. To allow read & write permissions for the user *lisa* we can use : `u:lisa:rw`.

Default ACL

After setting any ACL we also need to set up a default ACL that'll handle all items that we're going to create later in the future in that folder. This is done by specifying a `d` (default) in the `setfacl` command:

```
$ setfact -m d:g:account:rx /data/account
```

7.8 Managing ACLs

Let us consider a case where the account group needs read only access to the sales directory and vice versa. Of course we don't want to grant any access to others. Now, we need to assign a secondary group to the *sales* and *account* directory without removing their respective primary groups. This can be done using ACLs.

When the ACLs haven't been setup yet, the `getfacl` command shows the same information as the `ls -l` command.

```
# getfacl account
file: account
owner: lori
group: account
flags: -st
user::rwx
group::rwx
other::---
```

The `flags: -st` parameter shows us whether the SetUID, SetGID and Sticky Bit are set, in that order (sst). Since the GID is set, as is the sticky bit, but not the UID, the flags shows up as `-st`.

Note that the ACLs are copying over the current permission settings to the ACL. Thus, before setting ACLs, we need to ensure our permissions are exactly the way we want them to be. If we try to change the permission settings after creating the ACLs, we will end up in a mess.

Option		**Description**
-m	-	Modify, followed immediately by what needs to be modified.
-R	-	Recursive, i.e., apply to all files currently in the directory.

To set the sales group to have read access on the account folder and to check the permissions, we use:

```
# setfacl -R -m g:sales:r account
# getfacl account
file: account
owner: lori
group: account
flags: -st
user::rwx
group::rwx
group:sales:r--
mask::rwx
other::---
```

This only takes care of the items already present in the *account* directory, but not the new files that will be created in it. For that, we need to setup a default ACL. NOTE that default ACLs do no need to be applied recursively.

```
# setfacl -m d:g:sales:r account
getfacl account
file: account
owner: lori
group: account
flags: -st
user::rwx
group::rwx
group:sales:r--
mask::rwx
other::---
default:user::rwx
default:group::rwx
default:group:sales:r--
default:mask::rwx
default:other::---
```

The default ACL for the user, groups, etc are created from the current permission settings of the directory. If we make a directory in it, the following will be the ACL for it:

```
# cd account/
# mkdir 2017
# getfacl 2017
file: 2017
owner: root
group: account
```

```
flags: -s-
user::rwx
group::rwx
group:sales:r--
mask::rwx
other::---
default:user::rwx
default:group::rwx
default:group:sales:r--
default:mask::rwx
default:other::---
```

Now, if we were to make a new file in this directory, we get the following ACL for it: (Note that a file, by definition, can't have any default settings unlike directories, since they are leaf objects that can't have any children to apply the default permissions).

```
# cd 2017/
# touch testFile
# getfacl testFile
file: testFile
owner: root
group: account
user::rw-
group::rwx                          #effective:rw-
group:sales:r--
mask::rw-
other::---
```

Note that the mask has become active. This is because in case of files, we never want to grant execute permissions by default. So, even though in the POSIX permission, the group is granted `rwx` permission set, the mask of `rw-` is superimposed on it, and the union of the two, (i.e., `rw-`) is the effective permissions on the file for the owner group.

Thus, we need to remember that whenever we set ACLs on a directory we need two commands: one to set the ACL for the existing files, and the other for the default ACLs for the new files that can be created in the directory. Contrastingly, ACLs need to be set with only one command in case of files (when manually setting them to a file; inheritance of ACLs is automatic).

7.8.1 history

The `history` command shows us all the commands that were executed on the terminal (since last boot).

Chapter 8

Configuring Networking

8.1 Understanding NIC Naming

A Network Interface Card (NIC) is the physical hardware connecting the host machine to the network. In older versions of RHEL, the naming convention was simpler, with the Ethernet interface named simply as *eth0*. In RHEL 7, there are 3 different naming schemes available.

8.1.1 Network Device Naming Schemes

Scheme	Description	Example
BIOS Naming	Based on Hardware properties of the Network card.	**em[$1-N$]** Embedded NICs **p6p5** PCI Slot-6, Port-1
Udev Naming	Classical Naming Scheme with Interface number.	**Eth**N - example: eth0, eth1...
Physical Naming	Same as BIOS Naming	
Logical Naming	.<vlan> and :<alias>	

The entire purpose to the NIC naming scheme is to make it easier to identify the hardware NIC associated with an interface for cases where multiple NICs are available at a server.

8.2 Managing NIC Configuration with ip Command

An important feature of the `ip` command is all data is lost with interface is reset. However, it's extremely useful as it allows us to test certain settings on NICs. The syntax of the ip command is : `ip <options> <object> <command>`

Object	-	Description
link	-	Network status information
addr	-	Set network addresses
route	-	Helps manage routing table on the system

To get the help for any option and object, simply replace the command with the text "help". For example, to get help about the `ip addr` command, we need to type `ip addr help`.

8.2.1 show commands

ip link show

This command displays the device attributes, and can be followed by a device name to only view the details for that device/interface. Can be used to find the available interface names and the associated MAC addresses.

```
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
link/ether 00:0c:29:d6:73:d0 brd ff:ff:ff:ff:ff:ff
3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT qlen 1000
link/ether 52:54:00:a5:7f:97 brd ff:ff:ff:ff:ff:ff
4: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 state DOWN mode DEFAULT qlen 1000
link/ether 52:54:00:a5:7f:97 brd ff:ff:ff:ff:ff:ff
```

ip addr show

`ip addr show` command shows us the current (network) address information. Based on the interface name, we can find its specific information only as well, in which case the command has to be followed by the name of the interface.

```
$ ip addr show ens33
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
link/ether 00:0c:29:d6:73:d0 brd ff:ff:ff:ff:ff:ff
inet 90.0.16.117/21 brd 90.0.23.255 scope global dynamic ens33
valid_lft 3037sec preferred_lft 3037sec
inet6 fe80::2b85:fb69:3b97:ec5f/64 scope link
valid_lft forever preferred_lft forever
```

Note that the `inet` address is the IPv4 address whereas `inet6` refers to the IPv6 address.

ip route show

```
$ ip route show
default via 90.0.16.1 dev ens33 proto static metric 100
90.0.16.0/21 dev ens33 proto kernel scope link src 90.0.16.117 metric 100
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1
```

So, in this case, we have a default route which sends everything through the ip address *90.0.16.1*.

8.2.2 ip addr add

This command is used to add an IP address to a device. Note that while adding a new IP address, the address must always be followed by the Subnet Mask, as otherwise the default value of *42* is applied, which doesn't make sense.

```
# ip addr add dev ens33 10.0.0.10/24
# ip addr show ens33
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
link/ether 00:0c:29:75:5a:36 brd ff:ff:ff:ff:ff:ff
inet 90.0.18.206/21 brd 90.0.23.255 scope global dynamic ens33
valid_lft 3587sec preferred_lft 3587sec
inet 10.0.0.10/24 scope global ens33
valid_lft forever preferred_lft forever
inet6 fe80::ba08:1835:69e5:e9e9/64 scope link
valid_lft forever preferred_lft forever
```

This is one of the improvements of `ip` over `ifconfig` which was incapable of setting/showing multiple IP Addresses for the same device. `ifconfig` is obsolete. To see the network statistics (as shown in ifconfig command), the command is :

```
# ip -s link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
RX: bytes  packets  errors  dropped overrun mcast
55874      218      0       0       0       0
TX: bytes  packets  errors  dropped carrier collsns
55874      218      0       0       0       0
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
    DEFAULT qlen 1000
link/ether 00:0c:29:75:5a:36 brd ff:ff:ff:ff:ff:ff
RX: bytes  packets  errors  dropped overrun mcast
15408095   113357   0       0       0       0
TX: bytes  packets  errors  dropped carrier collsns
1857459    13898    0       0       0       0
```

8.2.3 ip route add

This command is used to add a new route.

```
# ip route add 20.0.0.0/8 via 192.168.4.4
```

However, all settings using the `ip` command are temporary and thus will be reset with every reboot. To makes these settings permanent, we need to provide this info in an appropriate configuration file to be loaded during each boot.

8.3 Storing Network Configuration persistently

The network configuration is stored in `/etc/sysconfig/network-scripts` directory. There are several configuration files for each interface. The configuration file for the *ens33* interface is called `ifcfg-ens33`. The script looks like:

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
```

```
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens33
UUID=1909bf04-c383-4aa0-afce-d774be49d3d4
DEVICE=ens33
ONBOOT=yes
HWADDR=00:0C:29:D6:73:D0
MACADDR=00:0C:29:D6:73:D0
IPADDR=192.168.4.44
PREFIX=24
GATEWAY=192.168.4.2
DNS1=8.8.8.8
```

The BOOTPROTO can be set to *dhcp* if DHCP is required. The *ONBOOT="yes"* sets that the NIC should be switched on during boot. The HWADDR property specifies the MAC address. IPADDR can be specified as IPADDR0 and thus more than one ip addresses can be specified as IPADDR1, IPADDR2, and so on.

8.3.1 Hostname

In the `/etc` directory, there is a file called **hostname** which contains the hostname of the machine we're working on. In earlier versions of RHEL, this information used to be stored in `/etc/sysconfig/network/`.

Another important file that isn't used anymore is `/etc/resolv.conf`. This file is auto-generated by the Network Manager, the most important part of the system that handles network configuration.

8.4 Understanding Network Manager

The Network Manager is the part of the OS that manages the NIC. There are three different ways of changing the network configuration on the NIC: using the `ip` command, the `nmcli` command and the Network Manager TUI (Text User Interface).

```
# ip addr add dev ens33 10.0.0.10/24
# nmcli con add con-name ens33 ifname ens33 type ethernet ip4 10.0.0.13/24
```

When the `ip addr add` command is used, the ip address is added directly to the physical NIC, which can start using it immediately. However, this data is impersistent since the information is not managed by any service. So, a reboot or even a simple bringing down of the interface erases the data. This is why the information needs to be stored in `/etc/sysconfig/network-scripts/ifcfg-ens33`.

So, when either the `nmcli` or the Network Manager TUI is used to configure the network settings, the Network Manager service ensures the data is stored in the above file and is thus available after every boot or interface restart.

The `ip` command is used for temporary changes only, while the Network Manager is used for persistent changes.

8.5 Using Network Manager utilities (nmcli, nmtui)

The `nmcli` tool controls the Network Manager from the command line. Just like the `ip` command, its syntax is `nmcli <options> <object> <command>`. The objects are like subcommands as in the case of `ip` command. The most important object is the **connection(c)**.

A network interface is just an interface with some connection associated with it. A connection is however, an abstract layer lying on top of the interface. The concept is that every Network interface has a default connection, but we can add a testing connection as well. We can then switch between these connections using the network manager utilities.

8.5.1 nmcli

nmcli connection show

```
$ nmcli connection show
NAME   UUID                                  TYPE            DEVICE
ens33  26c54678-6784-47ba-8afc-ca9924ed63af  802-3-ethernet  ens33
```

The command to add a new connection is: (line 9)

```
1  # ls /etc/sysconfig/network-scripts/
2  ifcfg-ens33  ifdown-eth   ifdown-post    ifdown-Team      ifup-aliases  ifup-ipv6
    ↪  ifup-post    ifup-Team      init.ipv6-global
3  ifcfg-lo     ifdown-ippp  ifdown-ppp     ifdown-TeamPort  ifup-bnep     ifup-isdn
    ↪  ifup-ppp     ifup-TeamPort  network-functions
4  ifdown       ifdown-ipv6  ifdown-routes  ifdown-tunnel    ifup-eth      ifup-plip
    ↪  ifup-routes  ifup-tunnel    network-functions-ipv6
5  ifdown-bnep  ifdown-isdn  ifdown-sit     ifup             ifup-ippp     ifup-plusb
    ↪  ifup-sit     ifup-wireless
6  # nmcli connection show
7  NAME   UUID                                  TYPE            DEVICE
8  ens33  26c54678-6784-47ba-8afc-ca9924ed63af  802-3-ethernet  ens33
9  # nmcli con add con-name testing ifname ens33 type ethernet ip4 10.0.0.15/24
10 Connection 'testing' (8bc5959e-3d1e-4738-8fa6-b584e4ba4388) successfully added.
11 # nmcli connection show
12 NAME     UUID                                  TYPE            DEVICE
13 ens33    26c54678-6784-47ba-8afc-ca9924ed63af  802-3-ethernet  ens33
14 testing  8bc5959e-3d1e-4738-8fa6-b584e4ba4388  802-3-ethernet  --
15 # ls /etc/sysconfig/network-scripts/
16 ifcfg-ens33    ifdown-bnep  ifdown-isdn    ifdown-sit       ifup          ifup-ippp
    ↪  ifup-plusb   ifup-sit       ifup-wireless
17 ifcfg-lo       ifdown-eth   ifdown-post    ifdown-Team      ifup-aliases  ifup-ipv6
    ↪  ifup-post    ifup-Team      init.ipv6-global
18 ifcfg-testing  ifdown-ippp  ifdown-ppp     ifdown-TeamPort  ifup-bnep     ifup-isdn
    ↪  ifup-ppp     ifup-TeamPort  network-functions
19 ifdown         ifdown-ipv6  ifdown-routes  ifdown-tunnel    ifup-eth      ifup-plip
    ↪  ifup-routes  ifup-tunnel    network-functions-ipv6
```

Note that upon creation of the new connection, the `nmcli` tool also creates a config file called `/etc/sysconfig/network-scripts/ifcfg-testing` (line 18) for the new connection called *testing*.

Switching Connections

The connection switching is as simple as bringing an interface down and bringing an alternative up. We do this using `nmcli` by:

```
# nmcli con show
NAME     UUID                                  TYPE            DEVICE
ens33    26c54678-6784-47ba-8afc-ca9924ed63af  802-3-ethernet  ens33
testing  8bc5959e-3d1e-4738-8fa6-b584e4ba4388  802-3-ethernet  --
# nmcli con down ens33
Connection 'ens33' successfully deactivated (D-Bus active path:
↪  /org/freedesktop/NetworkManager/ActiveConnection/11)
# nmcli con up testing
Connection successfully activated (D-Bus active path:
↪  /org/freedesktop/NetworkManager/ActiveConnection/13)
# nmcli con show
NAME     UUID                                  TYPE            DEVICE
testing  8bc5959e-3d1e-4738-8fa6-b584e4ba4388  802-3-ethernet  ens33
ens33    26c54678-6784-47ba-8afc-ca9924ed63af  802-3-ethernet  --
```

8.5.2 nmtui

This command provides a Text User Interface for the Network Manager.

On restarting the Network Manager, all the set connections become simultaneously activated. To avoid this, we would have to bring the connection down on restart either through the config files or the TUI.

```
# systemctl restart NetworkManager
# systemctl status -l NetworkManager
```

8.6 Understanding Routing and DNS

8.6.1 Default route

To connect to another network (or the internet), the server needs to know an IP Address of another computer that can connect it to the desired network. This is called the **default route**. It must be present on the same network as the host.

If we consider in the diagram that the Computer with the IP address *192.168.1.19* is our host that needs a packet to reach another computer on the internet, then the default route for it would be *192.68.1.1* as it is the computer through which our server is connected to another network. Further, to pass along the packet to the receiver on the internet, the computer with IP *192.168.1.1* will have to go through another computer's IP as it's own default route that can connect it to the internet. Here, that is *10.0.0.1*.

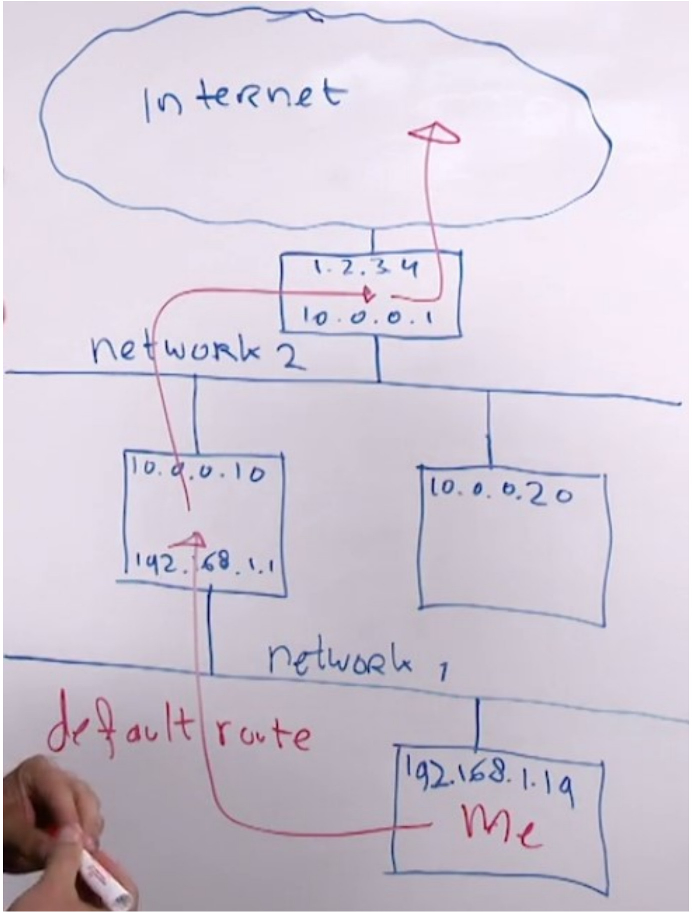

Figure 8.1: Default Route

Let us consider another scenario where the host *1*0.0.0.20 wants to send a packet to *192.168.1.19*. Its default route will have to through *192.168.1.1*.

Now, consider *10.0.0.20* needs to send a packet back to *192.168.1.19*, but also needs to send packets to the internet. Then, the default route would be the IP address of the computer that can connect us to the internet (*10.0.0.1*). However, to eventually reach *192.168.1.19*, the packets need a way to reach *192.168.1.1* first, given by *10.0.0.10*. Thus, sometimes a computer needs to be configured for multiple routers.

8.6.2 DNS

A Domain Name System (DNS) server stores the domain names along with a list of their corresponding IP addresses, and when packets destined for a certain domain name are available, it provides the actual IP addresses for it. It translates the domain name to an IP address.

8.7 Configuring Routing and DNS

On a temporary basis, the `ip route add` command can add a new default route. The settings can be shown using `ip route show`. However, to make the settings permanent, we need to edit the file `/etc/sysconfig/network-scripts/ifcfg-ens33` (where *ens33* is the name of our interface). Merely changing the value of the Gateway will suffice. After changing the value of the gateway, we need to bring the interface down and up again!

```
# nmcli con down ens33; nmcli con up ens33
Connection 'ens33' successfully deactivated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/3)
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/5)
```

Likewise, the values for DNS Server(s) are also specified in the `ifcfg-ens33` file. There can be multiple DNS servers, where the successive server(s) are contacted (in order) only if the preceding ones were down (or couldn't be contacted).

8.8 Understanding Network Analysis Tools

The following are a few network analysis tools that help diagnose problems with the network.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>hostname</td><td>Shows current hostname and provides an option to change it.</td></tr>
<tr><td>ping</td><td>Performs a connectivity test to know if another computer can be reached.</td></tr>
<tr><td>traceroute</td><td>Provides specific information about the routing between the host and a destination computer. NOTE that many routers are configured nowadays to not display information about their operation, and thus information from <code>traceroute</code> may be inaccurate.</td></tr>
<tr><td>dig</td><td>Shows DNS information and helps diagnose DNS related problems.</td></tr>
<tr><td>nmap</td><td>Advanced and potentially dangerous tool to get information about remote service availability. Since a portscan can be performed to determine which services are provided by a server using it, this utility is considered hostile by many NetAdmins.</td></tr>
<tr><td>netscan</td><td><table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>netstat -i</td><td>Packet information for network cards.</td></tr>
<tr><td>netstat -tulpen</td><td>Information about listening ports on a server:
t - TCP
u - UDP
l - Listening
p - Process Information
e - Extended Information
n - Names</td></tr>
</table></td></tr>
</table>

8.9 Using Network Analysis Tools

For troubleshooting network issues, we first check our own network information. We use `ip addr show` to ensure our IP address is correct. Then, we use `ip route show` to ensure that the default route is set to an IP that's in the same IP network as (one of) our own IP address. Next we check the DNS name resolution by printing `/etc/resolv.conf`.

```
# ip addr show
# ip route show
# cat /etc/resolv.conf
```

8.9.1 ping

If the problem still persists, further testing is required. First we ping the nearest router, then the one after that till we can reach the internet, unless there's an error, in which case we can know exactly which network has a problem.

```
# ping -c 1 192.168.0.1 # Pinging the default router by sending 1 packet.
```

ping flood test

This is a bandwidth test, instead of a connectivity test and tells us how many packets are being dropped on real time.

```
# ping -f cliServer
PING cliServer.somuVMnet.local (90.0.18.206) 56(84) bytes of data.
.^
--- cliServer.somuVMnet.local ping statistics ---
5215 packets transmitted, 5215 received, 0% packet loss, time 3354ms
rtt min/avg/max/mdev = 0.140/0.381/6.757/0.516 ms, ipg/ewma 0.643/0.287 ms
```

It prints a `.` for every ECHO_REQUEST and prints a backspace for every reply. Thus, the frequency of appearance of `.`s on the screen represents the frequency of packet loss. Further, since an interval is not given, it sends the packets as soon as a reply is received, thus giving us a measure of the bandwidth of the line in use. NOTE that superuser privileges are required to flood ping without an interval.

8.9.2 traceroute

En-route to the destination server, the time it took to reach every router and the time taken is displayed by this command. If there is some kind of filtering enabled on the server, then the data is redacted with `*`s.

```
# traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
1  amontpellier-653-1-352-1.w90-0.abo.wanadoo.fr (90.0.16.1)  4.277 ms  4.078 ms  3.954
↪  ms
2  202.38.180.121 (202.38.180.121)  7.041 ms  6.943 ms  6.890 ms
3  10.10.50.1 (10.10.50.1)  6.796 ms  6.565 ms  6.614 ms
4  202.38.180.50 (202.38.180.50)  12.069 ms  11.962 ms  11.829 ms
5  108.170.253.97 (108.170.253.97)  18.954 ms  18.886 ms 108.170.253.113
↪  (108.170.253.113)  18.768 ms
6  209.85.240.133 (209.85.240.133)  18.232 ms 209.85.240.159 (209.85.240.159)  12.603 ms
↪  72.14.239.7 (72.14.239.7)  12.381 ms
7  google-public-dns-a.google.com (8.8.8.8)  12.223 ms  15.113 ms  14.561 ms
```

8.9.3 host

The host command returns the IP address of any domain name that the machine can resolve with the host-name resolution process. This means a result will be produced even if the host is merely added in the `/etc/hosts` file and not configured on the DNS server.

```
# host www.somusysadmin.com
www.somusysadmin.com has address 104.27.137.245
www.somusysadmin.com has address 104.27.136.245
www.somusysadmin.com has IPv6 address 2400:cb00:2048:1::681b:89f5
www.somusysadmin.com has IPv6 address 2400:cb00:2048:1::681b:88f5
```

8.9.4 dig

dig gives more in-depth information about the name lookup process through DNS. This means no information is provided if the destination machine is unknown to the DNS server.

```
# dig www.somusysadmin.com

; <<>> DiG 9.9.4-RedHat-9.9.4-51.el7 <<>> www.somusysadmin.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65365
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 13, ADDITIONAL: 12

;; QUESTION SECTION:
;www.somusysadmin.com.                IN        A

;; ANSWER SECTION:
www.somusysadmin.com.        199        IN        A        104.27.136.245
www.somusysadmin.com.        199        IN        A        104.27.137.245

;; AUTHORITY SECTION:
com.                        100319        IN        NS        a.gtld-servers.net.
com.                        100319        IN        NS        j.gtld-servers.net.
com.                        100319        IN        NS        c.gtld-servers.net.
com.                        100319        IN        NS        m.gtld-servers.net.
com.                        100319        IN        NS        l.gtld-servers.net.
com.                        100319        IN        NS        h.gtld-servers.net.
com.                        100319        IN        NS        b.gtld-servers.net.
com.                        100319        IN        NS        d.gtld-servers.net.
com.                        100319        IN        NS        i.gtld-servers.net.
com.                        100319        IN        NS        g.gtld-servers.net.
com.                        100319        IN        NS        k.gtld-servers.net.
com.                        100319        IN        NS        f.gtld-servers.net.
com.                        100319        IN        NS        e.gtld-servers.net.

;; ADDITIONAL SECTION:
a.gtld-servers.net.        34680        IN        A        192.5.6.30
j.gtld-servers.net.        97045        IN        A        192.48.79.30
c.gtld-servers.net.        88438        IN        A        192.26.92.30
m.gtld-servers.net.        88436        IN        A        192.55.83.30
l.gtld-servers.net.        32537        IN        A        192.41.162.30
h.gtld-servers.net.        88436        IN        A        192.54.112.30
b.gtld-servers.net.        88438        IN        A        192.33.14.30
d.gtld-servers.net.        48310        IN        A        192.31.80.30
i.gtld-servers.net.        99400        IN        A        192.43.172.30
g.gtld-servers.net.        88436        IN        A        192.42.93.30
f.gtld-servers.net.        143261        IN        A        192.35.51.30
e.gtld-servers.net.        138671        IN        A        192.12.94.30

;; Query time: 766 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
```

```
;; WHEN: Wed Nov 29 11:59:42 IST 2017
;; MSG SIZE  rcvd: 486
```

The status of NOERROR indicates the operation was successful. The question section containing `www.somusysadmin.com. IN A` indicates that an address for `www.somusysadmin.com` was queried and the A indicates an address was asked for.

The answer section is provided with the different IP Addresses for the domain name. At the bottom, the server that was queried and operation details are noted.

`SERVER: 8.8.8.8#53(8.8.8.8).`

To perform a bit of performance optimization, we can add our own name server before a public DNS if we want to directly fetch the data. To do this, simply add a new DNS in `/etc/sysconfig/network-scripts/ifcfg-ens33`. After this, a restart of the NetworkManager is required for the new configuration settings to take effect. f

```
# systemctl restart NetworkManager
```

When we try to dig a site that doesn't exist, the output is :

```
# dig siteDoesntExist.com

; <<>> DiG 9.9.4-RedHat-9.9.4-51.el7 <<>> siteDoesntExist.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 22389
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;sitedoesntexist.com.                IN        A

;; Query time: 2 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Wed Nov 29 12:27:15 IST 2017
;; MSG SIZE  rcvd: 37
```

The NXDOMAIN status is indicative of the fact that the domain is non-existent.

8.9.5 Physical network problems

Sometimes there may be a physical problem in the network and not a problem with the configuration. In that case, the `ip -s link` command can give us a hint about the statistics of the interface and thus show us if packets are being dropped, etc.

```
# ip -s link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
RX: bytes  packets  errors  dropped overrun mcast
126271     1008     0       0       0       0
TX: bytes  packets  errors  dropped carrier collsns
126271     1008     0       0       0       0
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
 ↪  DEFAULT qlen 1000
link/ether 00:0c:29:d6:73:d0 brd ff:ff:ff:ff:ff:ff
RX: bytes  packets  errors  dropped overrun mcast
```

```
22610122   182819   0       0       0       0
TX: bytes  packets  errors  dropped carrier collsns
1686685    16970    0       0       0       0
3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode
 ↪  DEFAULT qlen 1000
link/ether 52:54:00:a5:7f:97 brd ff:ff:ff:ff:ff:ff
RX: bytes  packets  errors  dropped overrun mcast
0          0        0       0       0       0
TX: bytes  packets  errors  dropped carrier collsns
0          0        0       0       0       0
4: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 state DOWN
 ↪  mode DEFAULT qlen 1000
link/ether 52:54:00:a5:7f:97 brd ff:ff:ff:ff:ff:ff
RX: bytes  packets  errors  dropped overrun mcast
0          0        0       0       0       0
TX: bytes  packets  errors  dropped carrier collsns
0          0        0       0       0       0
```

Part II

Operating RHEL Servers

Chapter 9

Managing Processes

9.1 Understanding Jobs and Processes

During boot several services start up that in turn launch several processes that run in the background. They can be viewed by the `ps aux` command. Everything that's happening on a Linux system has a **process** behind it, which controls the actions. However, sometimes users launch a process from the shell - then it's called a **job**, related to that specific shell.

Let us consider a situation where a job keeps a shell busy. Consider the command is :

```
# dd if=/dev/zero of=/dev/null # Copies Nothing to Nowhere!
```

To stop the job from keeping the shell busy, we can move it to the background. The first step is to stop the job using *CTRL+Z*. Then, simply typing bg moves the job to the background!

9.1.1 jobs

The jobs command shows us an overview of all the jobs that are currently running.

```
$ dd if=/dev/zero of=/dev/null
^Z
[1]+  Stopped                 dd if=/dev/zero of=/dev/null
$ jobs
[1]+  Stopped                 dd if=/dev/zero of=/dev/null
$ bg
[1]+ dd if=/dev/zero of=/dev/null &
$ jobs
[1]+  Running                 dd if=/dev/zero of=/dev/null &
```

Jobs are tied to the current shell and user account. Any user will only see the jobs that were launched using the present shell when the `jobs` command is used.

9.2 Managing Shell Jobs

Every command on the shell is considered a shell job. Many of them start and stop immediately (execute very fast) because they've completed their task.

```
$ ls
Desktop    Downloads  Pictures  Public     Videos
Documents  Music      Programs  Templates
```

Other programs may need us to wait while they finish their jobs. In such cases, we can put them to work in the background and not have them obstruct our work.

```
$ cat test.sh
#!/bin/bash
echo "Starting"
sleep 10
echo "Completed!"
$ ./test.sh
Starting
^Z
[1]+  Stopped                 ./test.sh
$ jobs
[1]+  Stopped                 ./test.sh
$ bg
[1]+ ./test.sh &
$ Completed!

[1]+  Done                    ./test.sh
$ ./test.sh &
[1] 39773
Starting
$ Completed!

[1]+  Done                    ./test.sh
```

To directly start a job in the background, we need only suffix the command with a &. If we have multiple commands running in the background, we can put any one of them in the foreground using: fg <jobId>. Similarly, to kill a job, simply type: %<jobId>.

```
$ sleep 600 &
[1] 41461
$ dd if=/dev/zero of=/dev/null
^Z
[2]+  Stopped                 dd if=/dev/zero of=/dev/null
$ jobs
[1]-  Running                 sleep 600 &
[2]+  Stopped                 dd if=/dev/zero of=/dev/null
$ bg
[2]+ dd if=/dev/zero of=/dev/null &
$ jobs
[1]-  Running                 sleep 600 &
[2]+  Running                 dd if=/dev/zero of=/dev/null &
$ fg 1
sleep 600
^C
$ fg
dd if=/dev/zero of=/dev/null
^Z
[2]+  Stopped                 dd if=/dev/zero of=/dev/null
$ jobs
[2]+  Stopped                 dd if=/dev/zero of=/dev/null
$ kill %2
[2]+  Terminated              dd if=/dev/zero of=/dev/null
```

```
$ jobs
$
```

9.3 Getting process information with ps

- Shows a snapshot of all the running processes.
- `ps` shows only user's own processes.
- `ps aux` shows the processes of all users with the following details:

Option	Description
a	Show processes for all users
u	Display the process's user/owner
x	Also show processes not attached to a terminal

To see only the first 10 processes, use:

```
# ps aux | head
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.3 128164  6852 ?        Ss   Nov28   0:16 /usr/lib/systemd/systemd --switched-root --system --deserialize 21
root          2  0.0  0.0      0     0 ?        S    Nov28   0:00 [kthreadd]
root          3  0.0  0.0      0     0 ?        S    Nov28   0:09 [ksoftirqd/0]
root          5  0.0  0.0      0     0 ?        S<   Nov28   0:00 [kworker/0:0H]
root          7  0.0  0.0      0     0 ?        S    Nov28   0:00 [migration/0]
root          8  0.0  0.0      0     0 ?        S    Nov28   0:00 [rcu_bh]
root          9  0.0  0.0      0     0 ?        R    Nov28   0:42 [rcu_sched]
root         10  0.0  0.0      0     0 ?        S    Nov28   0:02 [watchdog/0]
root         12  0.0  0.0      0     0 ?        S    Nov28   0:00 [kdevtmpfs]
```

- The **PID** is the Process ID assigned to each process automatically by the Kernel. The PID 1 process is Systemd, which is the first process started by the kernel, which in turn starts all other processes.
- **%CPU** and **%MEM** are the CPU and Memory usage stats. *VSZ* is the Virtual Memory (total memory that the process has access to) while Resident Set Size (*RSS*) refers to the Physical Memory being used by the process. *Note that this includes the dynamic libraries, so if a library is used by a module multiple times, the memory used by the process will be smaller than the RSS*.
- **TTY** represents the terminal number on which the process is running. For background process, the TTY will be shown as `?`.
- **STAT** is the status of the process. *S* indicates the process is asleep.
- **TIME** is the total runtime of the process.
- **COMMAND** is the command that was executed.

Normally, we use `ps aux` to get the PID of a process. We can see the terminal grep itself is running on is set to `pts/0`, which is the gnome terminal.

9.3.1 Getting PID of a process

To see the PID of a particular process we simply grep some keyword related to the process, typically the process name.

```
# ps aux | grep packagekitd
root       1680  0.0  0.3 480016  5752 ?        Ssl  Nov28   0:05 /usr/libexec/packagekitd
root      44081  0.0  0.0 112660   976 pts/0    R+   21:04   0:00 grep --color=auto packagekitd
```

The `R` status indicates that the command `grep` was running at the time.

w command

The `w` command shows all the logged in users, what they're doing and the consequent load on the system.

```
# w
20:58:00 up 2 days,  1:09,  2 users,  load average: 0.00, 0.01, 0.05
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
somu     :0       :0               Mon09   ?xdm?   1:12m  0.84s /usr/libexec/gnome-session-binary --session gnome-classic
somu     pts/0    :0               13:41    0.00s  0.67s 10.88s /usr/libexec/gnome-terminal-server
```

9.3.2 Seeing Parent and Child process relation

Sometimes we need to see the relation between the processes, because when we kill a parent process, the child processes are also killed automatically! For that we use the command `ps fax`.

```
# ps fax | tail
37204 ?        S      0:00  \_ gnome-pty-helper
37205 pts/0    Ss     0:00  \_ bash
43222 pts/0    S      0:00      \_ su -
43230 pts/0    S      0:00          \_ -bash
43291 pts/0    S      0:00              \_ su - somu
43292 pts/0    S      0:00                  \_ -bash
43465 pts/0    S      0:00                      \_ su
43471 pts/0    S      0:00                          \_ bash
44146 pts/0    R+     0:00                              \_ ps fax
44147 pts/0    D+     0:00                              \_ [tail]
```

9.4 Understanding Memory Usage

To get an overview of the free memory available we use the command `free -m` where `-m` stands for Mega-Bytes. *Free* is the unused physical RAM. *Shared* refers to the memory used by shared libraries (shared by different programs). *buff/cache* is the combined Buffer and/or cache usage, where buffer is typically used when there's a large amount of data that needs to be committed to disk. *Available* is the amount of memory available for starting new applications without swapping.

```
# free -m
total        used        free      shared  buff/cache   available
Mem:          1823         930         198          18         694         656
Swap:         1907           0        1907
```

Swap contains the unused memory pages from the RAM that have been transfered to the Hard disk.

9.5 Understanding Performance Load

When a process is ready for execution, it's added to the runqueue, where it awaits evaluation by the scheduler to be assigned to a CPU. The amount of processes awaiting to be executed determines the performance load.

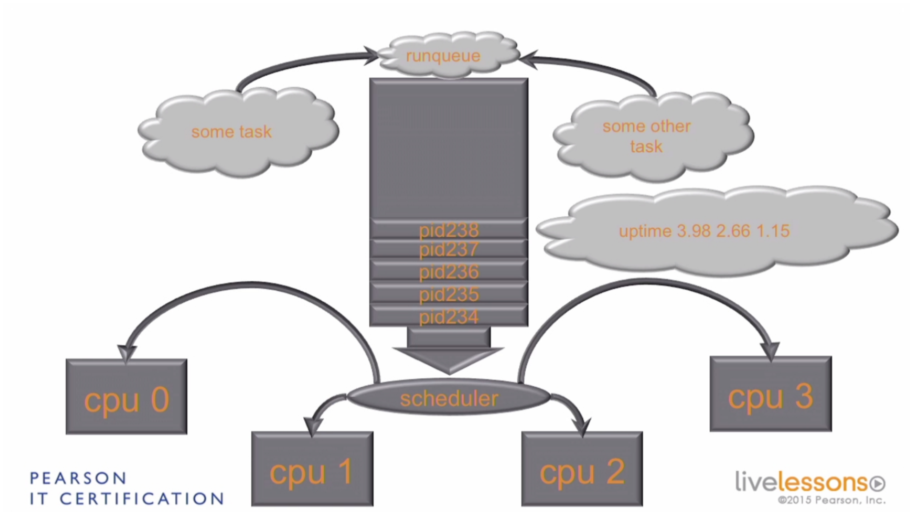

Figure 9.1: Performance Load

9.5.1 uptime Command

The `uptime` command shows us how long the system has been on, the number of users logged on and a snapshot of the current CPU demand for the last 1, 5 and 15 minutes, in that order. However, this number is represented for the total CPU resources being used, and not an average! Thus, for a single CPU system, uptime of 1 = 100% usage, while for a 4 CPU system, it means the CPU has been 75% idle.

The *System Load Average* is the number of processes in the runqueue that are either in runnable (ready to run/already running) state or in an interruptible state (e.g., waiting for I/O access).

```
# uptime
11:59:13 up 2 days,  6:02,  2 users,  load average: 0.00, 0.01, 0.05
```

The `nproc` command shows us the number of CPU cores available to us.

```
# nproc
1
```

9.6 Monitoring System Activity with top

top shows us the top active processes on the system in real-time.

```
# top
top - 12:24:09 up 2 days,  6:27,  2 users,  load average: 0.15, 0.07, 0.06
Tasks: 207 total,   2 running, 205 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.9 us,  0.7 sy,  0.0 ni, 95.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1867024 total,   196468 free,   957244 used,   713312 buff/cache
KiB Swap:  1953788 total,  1953788 free,        0 used.   667488 avail Mem

PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
2023 somu      20   0 1943348 246224  49292 S  5.3 13.2  61:53.62 gnome-shell
2250 somu      20   0  525584  12240   5984 S  2.3  0.7   0:00.61 tracker-store
1367 root      20   0  291800  35340  10312 S  1.3  1.9   2:53.91 X
48709 root      20   0  157716   2284   1536 R  0.7  0.1   0:00.06 top
764 root      20   0  305080   6140   4768 R  0.3  0.3   6:45.26 vmtoolsd
2270 somu      20   0  385944  19688  15236 S  0.3  1.1   6:02.21 vmtoolsd
2290 somu      39  19  637720  11280   8036 S  0.3  0.6   0:00.52 tracker-miner-f
48449 root      20   0       0      0      0 S  0.3  0.0   0:00.80 kworker/0:0
1 root      20   0  128164   6852   4084 S  0.0  0.4   0:18.42 systemd
2 root      20   0       0      0      0 S  0.0  0.0   0:00.22 kthreadd
3 root      20   0       0      0      0 S  0.0  0.0   0:11.45 ksoftirqd/0
5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
7 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0
8 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
9 root      20   0       0      0      0 S  0.0  0.0   0:48.31 rcu_sched
10 root      rt   0       0      0      0 S  0.0  0.0   0:02.52 watchdog/0
12 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kdevtmpfs
13 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 netns
14 root      20   0       0      0      0 S  0.0  0.0   0:00.22 khungtaskd
15 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 writeback
16 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kintegrityd
17 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 bioset
18 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kblockd
19 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 md
25 root      20   0       0      0      0 S  0.0  0.0   0:00.20 kswapd0
```

If we put 3 processes that all run `dd if=/dev/zero of=/dev/null` on the runqueue, and then execute the `top` command, and then press the `1` key, all the individual CPU loads are displayed. Since our VM has only one CPU, it shows up as *Cpu0*.

```
# top
...
%Cpu0  : 27.6 us, 72.4 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
...
PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
48739 root      20   0  107948    612    516 R 32.9  0.0   0:04.59 dd
48737 root      20   0  107948    612    516 R 32.2  0.0   0:06.47 dd
48738 root      20   0  107948    608    516 R 32.2  0.0   0:04.94 dd
```

The first line of the output of the `top` command is the same as that of the `uptime` command. The top command also shows the Free and available memory in both physical RAM as well as Swap.

Now since there is only one cpu available, the 3 busiest processes, (the `dd` commands) have evenly distributed the CPU cycles among them (assigned by the scheduler), nearly 33% each (the rest is overheads and cycles used by other processes).

```
%Cpu(s): 26.5 us, 73.5 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

The CPU line shows the CPU Usage stats in percentages. Here, we see the CPU usage summary [Cpu(s)], the *us* stands for user-space, or processes started by the user. The *sy* shows us the system space CPU Usage, and is typically programs that directly deal with hardware. Since we've used the `dd` command that's directly copying data from one device to another, we have a high system space CPU usage.

The *id* stands for idle and shows us the percentage of time the system is idle. The *wa* shows the percentage of time the system is waiting for I/O operation completion, such as a slow disk or network.

9.7 Sending Signals to processes

Signals are mandatory instructions that the process can't ignore. Some of the most common Signals are **SIGTERM** and **SIGKILL**. The `SIGTERM` signal asks a process to cease its activity. If the signal doesn't work, i.e., the process doesn't obey it, then we send the `SIGKILL` signal, which terminates the process.

We can send these signals by the use of the `top` command. While top is running, we have to press the `k` key to initiate signal sending via `kill`. To choose the appropriate process, we enter the PID. The default signal is `SIGTERM` (a.k.a. Signal 15). The process is terminated immediately on sending the signal.

When we send Signal 9 (`SIGKILL`) the process doesn't have time to save or clean up its work, and thus the execution stops instantaneously! This means if a process is working on an open file, the `SIGKILL` signal can cause irreparable damage to it.

9.7.1 kill command

To send a signal directly from the command line, we use the `kill` command. To send a `SIGTERM` signal, we merely provide the PID. To send a `SIGKILL` signal, we have to provide a signal number of `9`.

```
# ps aux | grep dd
...
root      49575 50.3  0.0 107948   608 pts/0    R    13:25
[2]+  Running                 dd if=/dev/zero of=/dev/null &
# kill -9 49575
# jobs
[2]+  Killed                  dd if=/dev/zero of=/dev/null
```

To kill all processes matching a certain process using Signals uses:

```
# top
...
PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
49747 root      20   0  107948    608    516 R 22.8  0.0   0:05.81 dd
49748 root      20   0  107948    612    516 R 22.5  0.0   0:05.30 dd
49749 root      20   0  107948    612    516 R 22.5  0.0   0:04.91 dd
49746 root      20   0  107948    612    516 R 22.2  0.0   0:06.60 dd
...
# jobs
[1]   Running                 dd if=/dev/zero of=/dev/null \&
```

```
[2]   Running                 dd if=/dev/zero of=/dev/null \&
[3]-  Running                 dd if=/dev/zero of=/dev/null \&
[4]+  Running                 dd if=/dev/zero of=/dev/null \&
# killall dd
# jobs
[1]   Terminated              dd if=/dev/zero of=/dev/null
[2]   Terminated              dd if=/dev/zero of=/dev/null
[3]-  Terminated              dd if=/dev/zero of=/dev/null
[4]+  Terminated              dd if=/dev/zero of=/dev/null
```

9.8 Understanding Priorities and Niceness

Consider a hypothetical scenario where a bunch of people are standing in a queue to get movie tickets. There, everyone in the queue has the same priority. Now, if a lady comes along and skips the queue, we can fairly say she isn't nice. The same goes for processes in a Linux system.

The processes are born with the same priority, but their niceness can be adjusted. If the niceness is negative, the process is served first, and if positive, it lets the other processes (with lower niceness) be served before itself!

9.9 Changing Process Nice values

In the output of the `top` command, the *PR* column shows us the priority of each individual process. Normal processes are started with the same priority of **20**. There are certain real-time processes as well, that have a value of `rt` in their priority column.

In case we need to increase or decrease the priority of a process (e.g., higher priority to quickly complete a query, or free resources for other users by lowering priority) we adjust the niceness (*NI*) value of the process. This process is called *nicing* a process.

The niceness of a process can range from -20 to $+19$, and the consequent value of priority is related as :

$$PR = 20 + NI$$

Thus, the priority of a process can range from 0 (most aggressive) to 39 (nicest). We can adjust the niceness in small increments instead of huge jumps (setting niceness to -20) and see if that does our work. If not, we can change the niceness incrementally till our goals are met!

9.9.1 Chaning niceness from top

To renice a process from top, we can simply press `r`, then select the PID to renice and enter the new niceness value.

9.9.2 Changing Niceness from command line

We can also set the niceness from the command line, both while first running the process using the `nice` command, or change the nice value of a running process using the `renice` command.

```
# nice -n 10 dd if=/dev/zero of=/dev/null &
# top | head
top - 15:40:26 up 2 days,  9:44,  3 users,  load average: 1.10, 0.70, 0.71
Tasks: 215 total,   3 running, 212 sleeping,   0 stopped,   0 zombie
%Cpu(s):  9.5 us, 71.4 sy, 19.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1867024 total,   233356 free,   919716 used,   713952 buff/cache
KiB Swap:  1953788 total,  1953788 free,        0 used.   704744 avail Mem

PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
51465 root      30  10  107948    612    516 R 75.0  0.0   2:00.96 dd
1367 root      20   0  290560  34108  10412 S  5.0  1.8   3:22.26 X
2023 somu      20   0 1923992 226836  49300 S  5.0 12.1  66:10.12 gnome-shell
# renice -n -10 51465
51465 (process ID) old priority 10, new priority -10
# top | head
top - 15:42:18 up 2 days,  9:45,  3 users,  load average: 1.41, 0.92, 0.79
Tasks: 215 total,   2 running, 213 sleeping,   0 stopped,   0 zombie
%Cpu(s): 25.0 us, 75.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1867024 total,   233440 free,   919632 used,   713952 buff/cache
KiB Swap:  1953788 total,  1953788 free,        0 used.   704828 avail Mem

PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
51465 root      10 -10  107948    612    516 R 64.0  0.0   3:47.16 dd
1 root      20   0  128164   6852   4084 S  0.0  0.4   0:19.34 systemd
2 root      20   0       0      0      0 S  0.0  0.0   0:00.23 kthreadd
```

If however we find that we have to renice processes all the time, we might want to move some resource-hungry processes to other server(s).

Chapter 10

Managing Software

10.1 Understanding Meta Package Handlers

In the old days, the `rpm` command was used to install packages, and it was incapable of resolving dependencies (i.e., auto-installing other packages/programs that were needed to make a package work). The syntax needed for rpm is : `rpm -ivh packageName.rpm`. (**i**=install, **v**=verbose, **h**=show hashes about progress).

Nowadays, due to the `yum` package installer, this is no longer an issue. It works with repositories, which are installation sources for a bunch of packages, and the command works by downloading indexes for the repositories. The `yum` meta-package handler needs only the rpm name to install it.

```
# yum install blah.rpm
```

At this point the yum command searches the indexes for any dependencies. If any are found, they're downloaded from the repository as well, before the original package is installed.

10.2 Setting up Yum repositories

A standard RHEL installation is hooked up to RHN (Red Hat Network), the RedHat repository, and all patches and updates are downloaded from it. It's the installation source and primary repository for most packages available on RHEL.

10.2.1 yum repolist

This command shows us the list of repositories which our system is configured to use. Unless RHN is connected to, the RHEL 7 Server can't use this (and other) repo commands.

10.2.2 Custom Repository

To convert an existing folder to a yum repository, we need to first go to the `/etc/yum.repos.d` directory, and then create a file named: `repoName.repo` where *repoName* is the name of our custom repository. It's critical that the file name ends with `.repo` as otherwise yum won't be able to recognize it. The contents of the repoName.repo file should be:

```
[repoName]
name=repoName
baseurl=file:///home/somu/repo
gpgcheck=0
```

The first line is called the label. The second line defines the name of the repository. The third line, defines the URI (Uniform Resource Identifier) where the repository is located. If it's on the internet, protocols such as `ftp://` can be used, but in our case since it's on the local filesystem, we use the `file://` protocol. Further, the path of the repository folder is `/home/somu/repo`, which is what the *baseurl* is set to. The fourth line turns off the GPG file integrity checking (not suggested for real environments).

createrepo

A final step is to generate the indexes required by yum to use the repository. For this, we use the `createrepo` command.

```
# createrepo /downloads
Spawning worker 0 with 4 pkgs
Workers Finished
Saving Primary metadata
Saving file lists metadata
Saving other metadata
Generating sqlite DBs
Sqlite DBs complete
```

Next we can check if the repo was successfully added by running `yum repolist`.

```
# yum repolist
Loaded plugins: fastestmirror, langpacks
repo id                         repo name                         status
base/7/x86_64                   CentOS-7 - Base                   9,591
extras/7/x86_64                 CentOS-7 - Extras                   283
repoTestLabel                   repoTest                              0
updates/7/x86_64                CentOS-7 - Updates                1,134
repolist: 11,008
```

10.3 Using the yum command

The `yum` command is a package manager and a meta package handler. The following are some of the yum commands:

10.3.1 yum search

`yum search` searches the given repositories for a suitable package.

```
# yum search nmap
Loaded plugins: fastestmirror, langpacks
```

```
Loading mirror speeds from cached hostfile
* base: centos.excellmedia.net
* extras: centos.excellmedia.net
* updates: centos.excellmedia.net
======================== N/S matched: nmap ========================
nmap-frontend.noarch : The GTK+ front end for nmap
nmap-ncat.x86_64 : Nmap's Netcat replacement
nmap.x86_64 : Network exploration tool and security scanner

Name and summary matches only, use "search all" for everything.
```

10.3.2 yum install

`yum install` installs the package passed as argument to it, after installing all the required dependencies. When the `-y` option is used, Yum doesn't wait for a (Y/N) reply after showing the dependency list, and proceeds to download and install the package.

```
# yum install -y nmap
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
* base: centos.excellmedia.net
* extras: centos.excellmedia.net
* updates: centos.excellmedia.net
Resolving Dependencies
--> Running transaction check
---> Package nmap.x86_64 2:6.40-7.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
Package        Arch            Version                  Repository      Size
================================================================================
Installing:
nmap           x86_64          2:6.40-7.el7             base           4.0 M

Transaction Summary
================================================================================
Install  1 Package

Total download size: 4.0 M
Installed size: 16 M
Downloading packages:
No Presto metadata available for base
nmap-6.40-7.el7.x86_64.rpm                               | 4.0 MB   06:38
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : 2:nmap-6.40-7.el7.x86_64                                     1/1
Verifying  : 2:nmap-6.40-7.el7.x86_64                                     1/1

Installed:
nmap.x86_64 2:6.40-7.el7

Complete!
```

Some programs may have a script that needs to be run to setup and configure it. In such cases, yum does it for us.

10.3.3 yum list

The `yum list` command is used to list the packages installed on a system, filtered on a specific criteria.

Options	Description
yum list all	Lists all available and installed packages
yum list installed	Only list the installed packages
yum list avialable	Only list the available packages

10.3.4 yum provides

Sometimes we don't know which package to install. For example, if we want to install and use *semanage*, an important utility to set up SELinux, we have to use the `yum search semanage` command to find all the info about the packages that offer it.

```
# yum search semanage
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
* base: centos.excellmedia.net
* extras: centos.excellmedia.net
* updates: centos.excellmedia.net
=========================== N/S matched: semanage ============================
libsemanage-python.x86_64 : semanage python bindings for libsemanage
libsemanage.i686 : SELinux binary policy manipulation library
libsemanage.x86_64 : SELinux binary policy manipulation library
libsemanage-devel.i686 : Header files and libraries used to build policy
: manipulation tools
libsemanage-devel.x86_64 : Header files and libraries used to build policy
: manipulation tools
libsemanage-static.i686 : Static library used to build policy manipulation tools
libsemanage-static.x86_64 : Static library used to build policy manipulation
: tools

Name and summary matches only, use "search all" for everything.
```

The above are the results that contain the string '*semanage*' in their names/descriptions, but may not contain the semanage binary that we require. For such cases, where we know the name of the binary utility, but don't know which package contains it, we use the `yum provides` command. The `*/semanage` is used to indicate it needs to search some file pattern.

```
# yum provides */semanage
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
* base: centos.excellmedia.net
* extras: centos.excellmedia.net
* updates: centos.excellmedia.net
libsemanage-devel-2.5-8.el7.i686 : Header files and libraries used to build
: policy manipulation tools
Repo        : base
```

```
Matched from:
Filename    : /usr/include/semanage

libsemanage-devel-2.5-8.el7.x86_64 : Header files and libraries used to build
: policy manipulation tools
Repo        : base
Matched from:
Filename    : /usr/include/semanage

policycoreutils-python-2.5-17.1.el7.x86_64 : SELinux policy core python
: utilities
Repo        : base
Matched from:
Filename    : /usr/sbin/semanage
Filename    : /usr/share/bash-completion/completions/semanage

policycoreutils-python-2.5-17.1.el7.x86_64 : SELinux policy core python
: utilities
Repo        : @anaconda
Matched from:
Filename    : /usr/sbin/semanage
Filename    : /usr/share/bash-completion/completions/semanage
```

10.3.5 yum remove

`yum remove <packageName>` checks the system to see if any installed packages are dependent upon the package we're trying to remove. If so, it removes the specified package and the dependent packages, unless one (or more) of them are protected. For example, `yum remove bash` fails as it'd have to remove Systemd and yum packages since they are heavily dependent on bash! Again, any `yum remove` command requires a prompt to be answered, which can be bypassed with `yum remove -y`.

```
# yum remove -y nmap
Loaded plugins: fastestmirror, langpacks
Resolving Dependencies
--> Running transaction check
---> Package nmap.x86_64 2:6.40-7.el7 will be erased
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
Package        Arch            Version                  Repository       Size
================================================================================
Removing:
nmap           x86_64          2:6.40-7.el7             @base            16 M

Transaction Summary
================================================================================
Remove  1 Package

Installed size: 16 M
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Erasing    : 2:nmap-6.40-7.el7.x86_64                                    1/1
```

```
Verifying  : 2:nmap-6.40-7.el7.x86_64                           1/1

Removed:
nmap.x86_64 2:6.40-7.el7

Complete!
```

10.4 Using rpm queries

Any software installed on our RHEL Servers are tracked in an rpm database, which supports queries to find out status and other information about packages. Rpm queries are most useful for SysAdmins when we need to find out more information about a package or software. For example, if we need to find out how to configure a time synchronization service called `chronyd`, first we find out where it is located.

```
# which chronyd
/usr/sbin/chronyd
```

Now that we know where the binary for the chrony daemon is located, we perform an rpm query on it, to find out which package it comes from:

```
# rpm -qf /usr/sbin/chronyd         # Query the package owning <file>
chrony-3.1-2.el7.centos.x86_64
```

Now that we know what package it comes from, we can list everything that the package `chrony` contains:

```
# rpm -ql chrony        # Query list
/etc/NetworkManager/dispatcher.d/20-chrony
/etc/chrony.conf
/etc/chrony.keys
/etc/dhcp/dhclient.d/chrony.sh
/etc/logrotate.d/chrony
/etc/sysconfig/chronyd
/usr/bin/chronyc
/usr/lib/systemd/ntp-units.d/50-chronyd.list
/usr/lib/systemd/system/chrony-dnssrv@.service
/usr/lib/systemd/system/chrony-dnssrv@.timer
/usr/lib/systemd/system/chrony-wait.service
/usr/lib/systemd/system/chronyd.service
/usr/libexec/chrony-helper
/usr/sbin/chronyd
/usr/share/doc/chrony-3.1
/usr/share/doc/chrony-3.1/COPYING
/usr/share/doc/chrony-3.1/FAQ
/usr/share/doc/chrony-3.1/NEWS
/usr/share/doc/chrony-3.1/README
/usr/share/man/man1/chronyc.1.gz
/usr/share/man/man5/chrony.conf.5.gz
/usr/share/man/man8/chronyd.8.gz
/var/lib/chrony
/var/lib/chrony/drift
/var/lib/chrony/rtc
/var/log/chrony
```

To see only the configuration files, instead of all files related to the package, we use:

```
# rpm -qc chrony        # Query config
/etc/chrony.conf
/etc/chrony.keys
/etc/logrotate.d/chrony
/etc/sysconfig/chronyd
```

To find the documentation for the package, we use:

```
# rpm -qd chrony        # Query documentation
/usr/share/doc/chrony-3.1/COPYING
/usr/share/doc/chrony-3.1/FAQ
/usr/share/doc/chrony-3.1/NEWS
/usr/share/doc/chrony-3.1/README
/usr/share/man/man1/chronyc.1.gz
/usr/share/man/man5/chrony.conf.5.gz
/usr/share/man/man8/chronyd.8.gz
```

To view all packages installed on our system, we can use:

```
# rpm -qa         # Query all
```

This command is especially useful to find out which version of a package is installed.

```
# rpm -qa | grep openjdk
java-1.8.0-openjdk-headless-1.8.0.151-1.b12.el7_4.x86_64
java-1.8.0-openjdk-1.8.0.151-1.b12.el7_4.x86_64
```

Pre and Post install Scripts

Many packages include pre and post installation scripts that we may need to find out about. If that is the case, we can use:

```
# rpm -q --scripts java-1.8.0-openjdk
postinstall scriptlet (using /bin/sh):

update-desktop-database /usr/share/applications &> /dev/null || :
/bin/touch --no-create /usr/share/icons/hicolor &>/dev/null || :
exit 0
postuninstall scriptlet (using /bin/sh):

update-desktop-database /usr/share/applications &> /dev/null || :
if [ $1 -eq 0 ] ; then
/bin/touch --no-create /usr/share/icons/hicolor &>/dev/null
/usr/bin/gtk-update-icon-cache /usr/share/icons/hicolor &>/dev/null || :
fi
exit 0
posttrans scriptlet (using /bin/sh):

/usr/bin/gtk-update-icon-cache /usr/share/icons/hicolor &>/dev/null || :
```

This step become critical when working on a production server, especially for security purposes since installing a package requires administrative (root) privileges. If the package is from an unverified source, we should know what exactly the package installation script does before executing it.

For 3^{rd} party, downloaded packages, that we might not have installed yet, we need to use the `rpm -qp` command instead. Thus, to list the contents of said 3rd party package, we use:

```
# rpm -qpl <packageName>.rpm
# rpm -qp --scripts <packageName>.rpm
```

The second line shows us the scripts (pre and post install) that'll be used by the downloaded (and NOT yet installed) package.

10.4.1 Installing a local rpm file

To perform the installation of an rpm file that we've downloaded from the internet, and it's not in a repository, we use `yum localinstall`.

To download said rpm, we can use a tool like `wget <rpmURL>`.

```
# ls -l
total 4056
-rw-r--r--. 1 root root 4152356 Nov 25  2015 nmap-6.40-7.el7.x86_64.rpm
# yum localinstall nmap-6.40-7.el7.x86_64.rpm
Loaded plugins: fastestmirror, langpacks
Examining nmap-6.40-7.el7.x86_64.rpm: 2:nmap-6.40-7.el7.x86_64
Marking nmap-6.40-7.el7.x86_64.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package nmap.x86_64 2:6.40-7.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
Package    Arch         Version           Repository                     Size
================================================================================
Installing:
nmap       x86_64       2:6.40-7.el7      /nmap-6.40-7.el7.x86_64        16 M

Transaction Summary
================================================================================
Install  1 Package

Total size: 16 M
Installed size: 16 M
Is this ok [y/d/N]: y
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : 2:nmap-6.40-7.el7.x86_64                                    1/1
Verifying  : 2:nmap-6.40-7.el7.x86_64                                    1/1

Installed:
nmap.x86_64 2:6.40-7.el7
```

```

Complete!
```

10.4.2 repoquery

The repoquery is similar to the rpm query, but instead of querying an installed or not-yet-installed but locally available package, it directly queries the repositories, without even needing to download them! However, the `--scripts` option isn't yet supported by the command.

```
# repoquery -ql yp-tools
/usr/bin/ypcat
/usr/bin/ypchfn
/usr/bin/ypchsh
/usr/bin/ypmatch
/usr/bin/yppasswd
/usr/bin/ypwhich
/usr/sbin/yppoll
/usr/sbin/ypset
/usr/sbin/yptest
/usr/share/doc/yp-tools-2.14
/usr/share/doc/yp-tools-2.14/AUTHORS
/usr/share/doc/yp-tools-2.14/COPYING
/usr/share/doc/yp-tools-2.14/ChangeLog
/usr/share/doc/yp-tools-2.14/NEWS
/usr/share/doc/yp-tools-2.14/README
/usr/share/doc/yp-tools-2.14/THANKS
/usr/share/doc/yp-tools-2.14/TODO
/usr/share/doc/yp-tools-2.14/nsswitch.conf
/usr/share/locale/de/LC_MESSAGES/yp-tools.mo
/usr/share/locale/sv/LC_MESSAGES/yp-tools.mo
/usr/share/man/man1/ypcat.1.gz
/usr/share/man/man1/ypchfn.1.gz
/usr/share/man/man1/ypchsh.1.gz
/usr/share/man/man1/ypmatch.1.gz
/usr/share/man/man1/yppasswd.1.gz
/usr/share/man/man1/ypwhich.1.gz
/usr/share/man/man5/nicknames.5.gz
/usr/share/man/man8/yppoll.8.gz
/usr/share/man/man8/ypset.8.gz
/usr/share/man/man8/yptest.8.gz
/var/yp/nicknames
```

10.4.3 Displaying information about a package

`repoquery -qi <packageName>` can display information about the package.

```
# repoquery -qi awesum

Name        : awesum
Version     : 0.6.0
Release     : 1
Architecture: noarch
Size        : 150637
Packager    : Darren L. LaChausse <the_trapper@users.sourceforge.net>
Group       : Applications/Security
```

```
URL         : http://awesum.sf.net/
Repository  : Ex11Repo
Summary     : Awesum is an easy to use graphical checksum verifier.
Source      : awesum-0.6.0-1.src.rpm
Description :
Awesum is a graphical checksum verification utility. It is written in Python
and uses the PyGTK toolkit. Awesum is very easy to use and includes support
for both MD5 and SHA checksum algorithms. Unlike many checksum verification
utilities, Awesum features a progress bar which makes working with large files
(such as CD-ROM ISO images) much more bearable.
```

Chapter 11

Working with Virtual Machines

11.1 Introducing KVM Virtualization

There are some basic requirements for KVM Virtualization on a server:

11.1.1 CPU Virtualization Support

The CPU needs to be capable to support virtualization. This can be easily verified using the command:

```
# grep -E "vmx|svm" /proc/cpuinfo
```

The presence of the `vmx` flag is the indication that the CPU supports Intel's VT-x virtualization. However, for AMD processors, the equivalent is `svm` indicating the presence of AMD's SVM technology.

If the processor does in fact support virtualization, then the Linux Kernel can load the `kvm` and the `kvm-intel` or `kvm-amd` modules. On top of the kernel lies the **libvirtd** daemon, which allow us to manage the KVM virtualization. It can also communicate other virtualization as well, such as Linux containers.

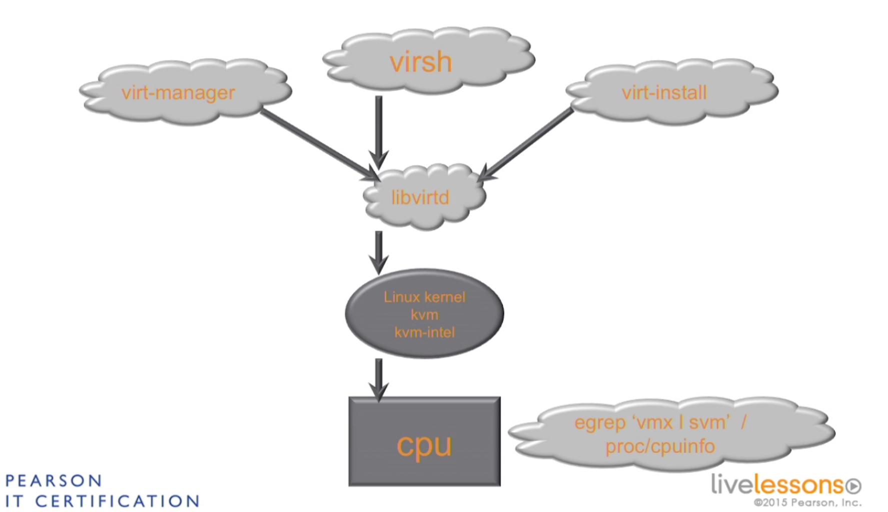

Figure 11.1: Virtualization

`libvirtd` only serves as a generic interface to virtualization that can be used by other management programs such as **virt-manager**, which is a GUI based VM management tool. There's also **virsh**, a shell-based VM manager, that is extremely useful to manage multiple VMs in an automated way!

There is also `vert-install`, a small installation interface that allow us to install VMs.

11.2 Managing Libvirt and KVM

First we need to verify that the required kernel modules for KVM are available. This can be done with:

```
# lsmod | grep kvm
kvm_intel             200704  0
kvm                   585728  1 kvm_intel
irqbypass              16384  1 kvm
```

The `lsmod` command shows us the status of the Linux Kernel modules. The `kvm` module is the generic KVM support module, while the `kvm-intel` module provides platform specific support for KVM virtualization.

Finally, we check if the `libvirtd` daemon is up and running using:

```
# systemctl status libvirtd
libvirtd.service - Virtualization daemon
Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; vendor pre
Active: active (running) since Sat 2017-12-02 16:08:39 IST; 2h 43min ago
Docs: man:libvirtd(8)
http://libvirt.org
Main PID: 945 (libvirtd)
Tasks: 18 (limit: 32768)
CGroup: /system.slice/libvirtd.service
945 /usr/sbin/libvirtd
1210 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default
1211 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default

Dec 02 16:08:55 lappyPrime dnsmasq[1210]: compile time options: IPv6 GNU-getopt
Dec 02 16:08:55 lappyPrime dnsmasq-dhcp[1210]: DHCP, IP range 192.168.124.2 -- 1
Dec 02 16:08:55 lappyPrime dnsmasq-dhcp[1210]: DHCP, sockets bound exclusively t
Dec 02 16:08:55 lappyPrime dnsmasq[1210]: no servers found in /etc/resolv.conf,
Dec 02 16:08:55 lappyPrime dnsmasq[1210]: read /etc/hosts - 2 addresses
Dec 02 16:08:55 lappyPrime dnsmasq[1210]: read /var/lib/libvirt/dnsmasq/default.
Dec 02 16:08:55 lappyPrime dnsmasq-dhcp[1210]: read /var/lib/libvirt/dnsmasq/def
Dec 02 16:09:37 lappyPrime dnsmasq[1210]: reading /etc/resolv.conf
Dec 02 16:09:37 lappyPrime dnsmasq[1210]: using nameserver 8.8.8.8#53
Dec 02 16:09:37 lappyPrime dnsmasq[1210]: using nameserver 202.38.180.7#53
```

Thus, this machine is completely ready for virtualization! If we run the `ip link show` command, we can also find a virtual bridge called virbr0, which is provided courtesy of KVM. This network acts as if it's connected to a virtual switch and provides inter-VM bridged networking support.

```
# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
 ↪  default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

```
2: enp1s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN mode DEFAULT group default qlen 1000
link/ether 3c:52:82:b9:05:5f brd ff:ff:ff:ff:ff:ff
3: wlo1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DORMANT group default qlen 1000
link/ether 3c:95:09:de:4e:8d brd ff:ff:ff:ff:ff:ff
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default qlen 1000
link/ether 52:54:00:62:77:f7 brd ff:ff:ff:ff:ff:ff
5: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel master virbr0 state DOWN mode DEFAULT group default qlen 1000
link/ether 52:54:00:62:77:f7 brd ff:ff:ff:ff:ff:ff
```

11.3 Using virsh

KVM requires a 64-bit architecture host. This can be verified by using:

```
# arch
x86_64
```

The `x86_64` output tells us that the installed OS is 64-bit. The Virtualization shell is opened by simply typing `virsh` which in turn supports a lot of commands.

```
# virsh
Welcome to virsh, the virtualization interactive terminal.

Type:  'help' for help with commands
'quit' to quit

virsh #
```

11.3.1 Virsh commands

virsh list

The following are the `virsh list` commands. They need to be run from the CLI directly. If the commands are being run from within the sub-shell provided by `virsh` then only the second command onward suffices (i.e., virsh need not be retyped).

Command	Description
virsh list	Shows all the virtual machines that are currently running
virsh list all	Shows all virtual machines that exist
virsh destroy <vmName>	Merely pulls the plug on the VM, i.e., immediately terminates the running VM.
virsh start <vmName>	Starts the VM.

Each virtual machine on the system has a configuration file that defines what the VM consists of. The `/etc/libvirt/` folder is related to the configuration of the `libvirtd` daemon that acts as a common interface for all KVM managers. Inside it is a directory called `qemu`. **qemu** is an old emulator that has been adapted for use in the KVM environment. The qemu configuration files are stored in this `/etc/libvirt/qemu` directory in XML format. This file is auto-generated by virsh, and thus should only be edited with `virsh edit <vmName>`. This has the benefit of ensuring nothing else is concurrently editing the file.

11.4 Using virt-manager

The `virt-manager` is a GUI interface for the control of VMs. It can be started by:

```
# virt-manager
```

Note that to control VMs made using `virt-manager` from within `virsh`, we need to be logged in under the same user when starting these utilities. They use the same libvirt database of VMs, but the user context can cause the VMs to be unavailable to some users!

Chapter 12

Scheduling Tasks

12.1 Cron vs at

Both `cron` and `at` enable us to perform a task in the future. However, cron lets us perform a job on a regular basis, while if we only need to perform a task once in the future, we use at.

12.1.1 Cron

Cron uses the `crond` service which is started by default and in turn used by many services. The configuration files for cron reside in various locations, and this allows RPMs to drop shell scripts in cron without any interruption or change of config. It also allows users to create their own cron jobs.

12.1.2 at

at which uses the `atd` daemon runs tasks only once in the future at a pre-scheduled time. The at command is used to add jobs.

12.2 Understanding Cron Configuration files and Execution times

The main configuration file for cron is located at `/etc/crontab`. The default contents of the file is:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
```

```
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

This config file lists the meaning of the time specifications for setting up a cronjob. We shouldn't modify this file as it'll be over-written every time the software is updated.

12.2.1 crontab -e

One of the recommended ways to create a new cronjob is to use the command `crontab -e`. Each user has his own *crontab* that contains instructions to the cron daemon to execute certain tasks at a certain time. The `crontab -e` command opens the user's crontab in editor mode and creates a cron config file for the present user.

12.2.2 Other cron config files

Many other cron config files are located in the `/etc` directory.

```
$ ls -ld cron*
drwxr-xr-x. 2 root root  54 Nov 25 08:59 cron.d
drwxr-xr-x. 2 root root  57 Nov 25 08:59 cron.daily
-rw-------. 1 root root   0 Aug  3 21:03 cron.deny
drwxr-xr-x. 2 root root  41 Nov 25 08:59 cron.hourly
drwxr-xr-x. 2 root root   6 Jun 10  2014 cron.monthly
-rw-r--r--. 1 root root 451 Jun 10  2014 crontab
drwxr-xr-x. 2 root root   6 Jun 10  2014 cron.weekly
```

Each of the folders named *cron.hourly*, *cron.daily*, *cron.weekly* and *cron.monthly* are used by RPMs to drop shell scripts that the cron daemon automatically executes on an appropriate schedule.

For example, the `/etc/cron.daily` has a file called *man-db.cron*. The `mandb` command (executed appropriately by this file) has to be executed periodically to rebuild the index that the man pages that we search using `man -k`. The file contains:

```
$ cat man-db.cron
#!/bin/bash

if [ -e /etc/sysconfig/man-db ]; then
. /etc/sysconfig/man-db
fi

if [ "$CRON" = "no" ]; then
exit 0
fi

renice +19 -p $$ >/dev/null 2>&1
ionice -c3 -p $$ >/dev/null 2>&1

LOCKFILE=/var/lock/man-db.lock

# the lockfile is not meant to be perfect, it's just in case the
# two man-db cron scripts get run close to each other to keep
# them from stepping on each other's toes.  The worst that will
```

```
# happen is that they will temporarily corrupt the database
[[ -f $LOCKFILE ]] && exit 0

trap "{ rm -f $LOCKFILE ; exit 0; }" EXIT
touch $LOCKFILE
# create/update the mandb database
mandb $OPTS

exit 0
```

The file is *not* a typical cronjob since we don't need to tell the cron daemon when to execute it. It knows that the given shell script has to be executed once daily. There is no fixed time for the cron job to run, and thus, even if the system goes down during a certain period of time, the cron daemon will execute the job at a later time.

12.2.3 cron.d

The files in this directory look a lot more like the crontab files. Some of the contents of this directory are:

```
$ ls -l /etc/cron.d
total 12
-rw-r--r--. 1 root root 128 Aug  3 21:03 0hourly
-rw-r--r--. 1 root root 108 Jun 13 19:38 raid-check
-rw-------. 1 root root 235 Aug  3 15:00 sysstat
```

The contents of the *sysstat* file is:

```
$ sudo cat sysstat
# Run system activity accounting tool every 10 minutes
*/10 * * * * root /usr/lib64/sa/sa1 1 1
# 0 * * * * root /usr/lib64/sa/sa1 600 6 &
# Generate a daily summary of process accounting at 23:53
53 23 * * * root /usr/lib64/sa/sa2 -A
```

The format followed is : time specification, name of the user under which the command has to be executed, followed by the command to be executed.

Thus, if a cronjob has to be run as an administrator, we should put it in a cron file in the `/etc/cron.d` directory. If however, a user-specific cron file has to be executed, then it's better to use the `crontab -e` command to generate and store the cronjobs for that user.

12.3 Scheduling with cron

One of the best ways to run cronjobs is to become the user that we want to run the cronjob as and then open their crontab, using the `crontab -e` command.

Let us consider a hypothetical scenario where we want to run a specific set of commands at 2.30PM everyday. Now, we first write the minutes(30) followed by the hour in 24-hour format [military time](2PM=14). Next, we want the script to run everyday, so we mark the day of the month, the month and the day of the week with *s (everyday of the month, every month and everyday of the week). Let us consider we want the cronjob to write something to the syslog using the `logger` command. Then the entry in the crontab will look like:

```
30 14 * * *        logger Hello
```

Typically, the `anacron` utility takes care of executing the shell scripts in the *cron.hourly, cron.daily, cron.weekly and cron.monthly* directories. It ensures that the commands will be executed at appropriate times (that cannot be controlled by the user) if the machine is down on the originally scheduled time.

12.4 Using at

Before using `at`, we need to ensure that the `atd` daemon is running:

```
$ systemctl status atd -l
atd.service - Job spooling tools
Loaded: loaded (/usr/lib/systemd/system/atd.service; enabled; vendor preset: enabled)
Active: active (running) since Sat 2017-12-02 15:26:18 IST; 1 day 8h ago
Main PID: 1224 (atd)
CGroup: /system.slice/atd.service
1224 /usr/sbin/atd -f

Dec 02 15:26:18 vmPrime.somuVMnet.local systemd[1]: Started Job spooling tools.
Dec 02 15:26:18 vmPrime.somuVMnet.local systemd[1]: Starting Job spooling tools.
```

12.4.1 Scheduling using at

The syntax for using `at` is simple: `at <time>`. This will open up the `at` prompt which takes as input the commands to be performed. We can escape the `at` prompt by sending an EOF signal using *CTRL+D*. We can schedule a message to be logged at 2:30PM using:

```
# at 1:17
at> logger Hello @ 02:30PM!
at> <EOT>
job 1 at Mon Dec  4 14:30:00 2017
```

Alternatively, the output of another command (or a shell script) can be piped to `at`.

```
$ at 00:38
at> echo "Hello from at @12:38AM" >> test.log
at> <EOT>
job 2 at Mon Dec  4 00:38:00 2017
$ date
Mon Dec  4 00:37:55 IST 2017
$ ls -l
total 0
$ date
Mon Dec  4 00:38:17 IST 2017
$ ls -l
total 4
-rw-rw-r--. 1 somu somu 23 Dec  4 00:38 test.log
$ cat test.log
Hello from at @12:38AM
```

12.4.2 atq

The `atq` command is used to see how many `at` jobs are waiting to be run.

```
$ echo 'echo "2mins after 1AM" >> time.log' | at 01:02
job 9 at Mon Dec  4 01:02:00 2017
$ echo 'echo "3mins after 1AM" >> time.log' | at 01:03
job 10 at Mon Dec  4 01:03:00 2017
$ ls -l
total 0
$ atq
9         Mon Dec  4 01:02:00 2017 a somu
10        Mon Dec  4 01:03:00 2017 a somu
$ date
Mon Dec  4 01:01:54 IST 2017
$ ls -l
total 0
$ date
Mon Dec  4 01:02:04 IST 2017
$ ls -l
total 4
-rw-rw-r--. 1 somu somu 16 Dec  4 01:02 time.log
$ cat time.log
2mins after 1AM
$ date
Mon Dec  4 01:03:01 IST 2017
$ cat time.log
2mins after 1AM
3mins after 1AM
```

12.4.3 Removing jobs from atq

The jobs scheduled by `at` can be removed by passing their job number to `atrm` command.

```
$ echo 'echo "Test" > file' | at 1:10
job 11 at Mon Dec  4 01:10:00 2017
$ echo 'echo "Test2" >> file' | at 01:11
job 12 at Mon Dec  4 01:11:00 2017
$ atq
11        Mon Dec  4 01:10:00 2017 a somu
12        Mon Dec  4 01:11:00 2017 a somu
$ atrm 11
$ atq
12        Mon Dec  4 01:11:00 2017 a somu
```

The `at` jobs are stored in a file in the `/var/spool/at` directory in a file with a system generated name.

The output of the `logger` command can be checked by using `tail -f /var/log/messages`.

```
# at 1:17
at> logger Hello @ 12:17AM!
at> <EOT>
job 15 at Mon Dec  4 01:17:00 2017
[root@vmPrime at]# atq
15        Mon Dec  4 01:17:00 2017 a root
```

```
# tail -f /var/log/messages
Dec  4 01:12:03 vmPrime dbus[702]: [system] Successfully activated service 'org.freedesktop.problems'
Dec  4 01:12:03 vmPrime dbus-daemon: dbus[702]: [system] Successfully activated service 'org.freedesktop.problems'
Dec  4 01:12:28 vmPrime journal: No devices in use, exit
Dec  4 01:17:00 vmPrime root: Hello @ 12:17AM!
```

Chapter 13

Configuring Logging

13.1 Understanding rsyslogd and journald logging

On RHEL 7 there are two systems responsible for logging : **rsyslogd** and **journald**. These two services together to handle logging system information.

It is up to the services (containing the logging information) to decide how and where the log files will be written to. It is possible to write directly to a log file anywhere (e.g., `/somewhere/my.log`). The service may also choose to pass over the information to **systemctl** as well. The `systemctl` utility is used to start the service and keep track of the actions of the service while it's starting. Anything that goes through **systemd** will be writing to `journald`, which is the systemd way of logging.

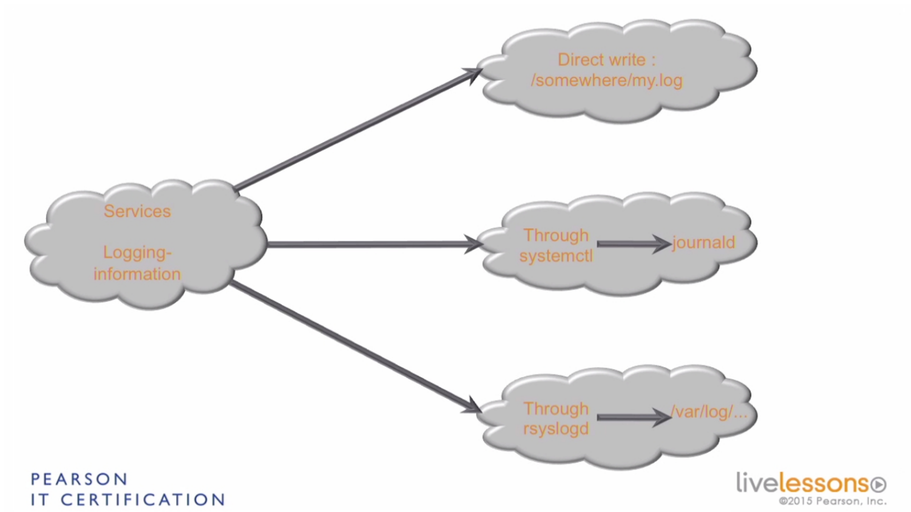

Figure 13.1: Logging Options

The classic way of logging is through the `rsyslogd` daemon, which typically writes information to the `/var/log/` directory. Alternative locations can also be used.

Given the various ways to log information, it may also become challenging for a user to get that logged information. It can be through **journalctl** or rsyslog. It is possible to tie these two systems together.

13.1.1 Sharing logging information

To ensure that journalctl information is automatically logged to rsyslog, we need to add a couple of lines to `/etc/rsyslog.conf` and `/etc/rsyslog.d/listend.conf`:

In **/etc/rsyslog.conf** :	In **/etc/rsyslog.d/listend.conf** :
`$ModLoad imuxsock` `$OmitLocalLogging off`	`$SystemLogSocketName` `↪ /run/systemd/journal/syslog`

The above lines enable rsyslog to receive information logged by journald. To enable the logging of rsyslog information in journald, we only need to add in the `/etc/rsyslog.conf`:

```
$Modload omjournal *.* :omjournal:
```

The above lines specify that any information being logged should be sent to `omjournal` which is a part of journald. The information from there is available from `journalctl`.

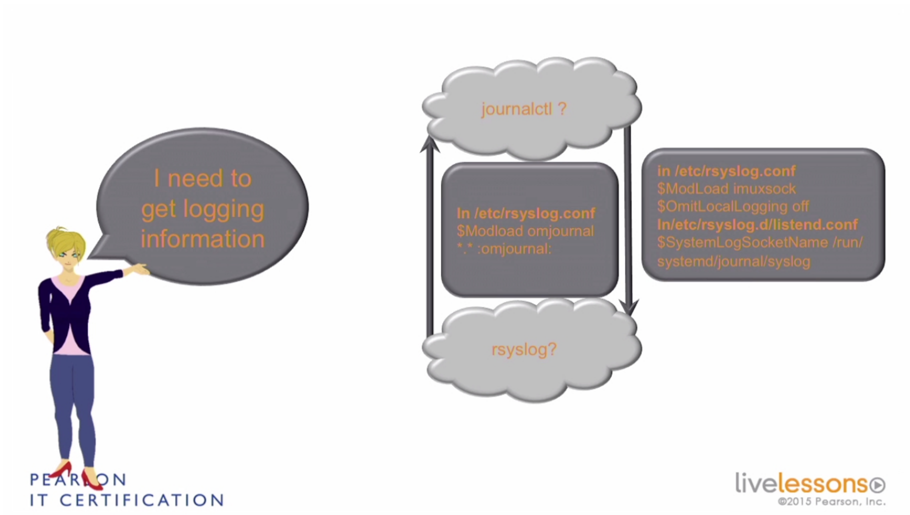

Figure 13.2: Sharing logging information between rsyslog and journald

13.2 Integrating rsyslogd and journald

Rsyslog is the old process of logging system information, and journald is the new process that's trying to do the same. Both of them are built to log process information.

13.2.1 rsyslog

Let us consider a process that's trying to write what it's doing in log files. It has two options: rsyslog and journald. If it writes to rsyslog, the advantage is that rsyslog is the system process that can handle logging for all processes. Some processes, however, just have some internal logging option (direct writes) thus bypassing rsyslog. A well known example of such a process is the *apache web server*. This is a disadvantage as if every process is doing its own logging, it's harder to manage the whole thing in a centralized way. These processes can however, be configured to write to rsyslog anyway!

13.2.2 journald

Journald is controlled by **systemd**, which takes care of starting services while booting. Systemd writes its own information to journald, and thus, if something goes wrong while starting a service, the information is available from *journald*.

For RHEL 7 in general, *journald* generally takes care of loggin the startup information logging while *rsyslog* takes care of logging current activity of processes.

13.3 Configuring rsyslog logging

[VIDEO TUTORIAL MISSING]

13.4 Working with journald

Everything that *journald* is doing is written to a binary file. The file can be explored using two methods: by using **systemctl** and **journalctl**. Journald integrates very well with systemctl and thus, systemctl commands can get information from journald and vice versa. Using `systemctl status <serviceName>` gives us the log information that systemctl receives from the journald environment.

13.4.1 journalctl

The `journalctl` command opens up the binary file that journald is writing to. Since it's a large file, there are several filtering options:

journalctl -b

The `journal -b` command only shows us the boot log.

```
# journalctl -b
-- Logs begin at Sat 2017-12-02 15:25:40 IST, end at Mon 2017-12-04 13:23:04 IST. --
Dec 02 15:25:40 vmPrime.somuVMnet.com systemd-journal[86]: Runtime journal is using 8.0M
↪  (max allowed 289.5M, trying to leave 434.3M free of 2.
Dec 02 15:25:40 vmPrime.somuVMnet.com kernel: Initializing cgroup subsys cpuset
Dec 02 15:25:40 vmPrime.somuVMnet.com kernel: Initializing cgroup subsys cpu
Dec 02 15:25:40 vmPrime.somuVMnet.com kernel: Initializing cgroup subsys cpuacct
Dec 02 15:25:40 vmPrime.somuVMnet.com kernel: Linux version 3.10.0-693.5.2.el7.x86_64
↪  (builder@kbuilder.dev.centos.org) (gcc version 4.8.5 2015
Dec 02 15:25:40 vmPrime.somuVMnet.com kernel: Command line:
↪  BOOT_IMAGE=/vmlinuz-3.10.0-693.5.2.el7.x86_64 root=/dev/mapper/centos-root ro crash
Dec 02 15:25:40 vmPrime.somuVMnet.com kernel: Disabled fast string operations
...
```

One of the best features of journald is that systemd initiates it immediately during boot and thus logs about what happens even during the very first stages of RHEL boot is available.

journalctl –since=<time>

The journalctl has a method to filter all results to show us what has happened since a specified period where only the logs written after a certain period are show.

```
# journalctl --since=yesterday
-- Logs begin at Sat 2017-12-02 15:25:40 IST, end at Mon 2017-12-04 13:28:03 IST. --
Dec 03 23:04:36 vmPrime.somuVMnet.local systemd[1]: Time has been changed
Dec 03 23:04:36 vmPrime.somuVMnet.local NetworkManager[834]: <info>  [1512322476.5889]
 ↪  audit: op="sleep-control" arg="off" pid=3311 uid=0 resul
Dec 03 23:04:36 vmPrime.somuVMnet.local systemd[1]: Stopping LSB: Bring up/down
 ↪  networking...
...
```

journald -u

The `journal -u` command shows us all the logs corresponding to a certain process.

```
# systemctl status atd -l
atd.service - Job spooling tools
Loaded: loaded (/usr/lib/systemd/system/atd.service; enabled; vendor preset: enabled)
Active: active (running) since Sat 2017-12-02 15:26:18 IST; 1 day 22h ago
Main PID: 1224 (atd)
CGroup: /system.slice/atd.service
1224 /usr/sbin/atd -f

Dec 02 15:26:18 vmPrime.somuVMnet.local systemd[1]: Started Job spooling tools.
Dec 02 15:26:18 vmPrime.somuVMnet.local systemd[1]: Starting Job spooling tools...
# journalctl -u atd
-- Logs begin at Sat 2017-12-02 15:25:40 IST, end at Mon 2017-12-04 13:37:28 IST. --
Dec 02 15:26:18 vmPrime.somuVMnet.local systemd[1]: Started Job spooling tools.
Dec 02 15:26:18 vmPrime.somuVMnet.local systemd[1]: Starting Job spooling tools...
```

Both systemctl and journald are intimately interconnected. Journald receives it's original logging information from systemctl, while the information displayed by the `systemctl status` command is derived from the information stored by journald. To see the information in even more detail, we use the command `journalctl -u <processName> -o verbose`.

```
# journalctl -u atd -o verbose
-- Logs begin at Sat 2017-12-02 15:25:40 IST, end at Mon 2017-12-04 13:44:14 IST. --
Sat 2017-12-02 15:26:18.215379 IST
 ↪  [s=7e5f5839...6e;i=8f2;b=f7106...c3;m=25810b8;t=55f58..dd;x=5e9c...46]
PRIORITY=6
_UID=0
_GID=0
_BOOT_ID=f7106b6e5bc144bcac5827c5089f23c3
_MACHINE_ID=9d29aa554cfa4853b59f2d517a8470bd
SYSLOG_FACILITY=3
SYSLOG_IDENTIFIER=systemd
...
```

The above command gives us complete information about the environment of the process.

13.5 Understanding logrotate

Logrotate is a system that's used to ensure that logging doesn't fill the hard disk of the server. Logrotate ensures that after a specified amount of time, log files will be closed and new ones opened, and a backlog of a couple of log files will be kept. This is all done based on the assumption that old logs are useless beyond a certain age. This has a stark disadvantage of erasing logging data that may become useful at a later date.

A solution to this problem is the use of a log server, with sufficient hard disk space to keep about a year's worth of logs. The client machines can have logrotate setup to keep only a couple of week's data since there will be a backup available on the log server.

13.6 Configuring logrotate

The configuration files for `logrotate` reside in the `/etc` directory. The generic logrotate configuration is stored in `/etc/logrotate.conf` while the directory `/etc/logrotate.d` contains include files that RPMs dump in it for package specific log rotation. Anything in the `logrotate.d` directory will always overwrite the settings in `logrotate.conf`. Typical contents of the `logrotate.conf` file is:

```
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# use date as a suffix of the rotated file
dateext

# uncomment this if you want your log files compressed
#compress

# RPM packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp and btmp -- we'll rotate them here
/var/log/wtmp {
monthly
create 0664 root utmp
minsize 1M
rotate 1
}

/var/log/btmp {
missingok
monthly
create 0600 root utmp
rotate 1
}
```

The last two settings demonstrate how specific logrotation instructions can be given for specific files. For example, the `/var/log/wtmp` file has to be rotated monthly, and only 1 copy of the backlog is maintained.

Since logrotate doesn't need to run all the time, it doesn't itself run as a service, but as a cron job! The config file for logrotate cron job is `/etc/cron.daily/logrotate`.

13.6.1 Checking avialable hard disk space

The available hard disk space and the disk space occupied by a certain directory can be checked using these two commands:

```
# df -h
Filesystem               Size  Used Avail Use% Mounted on
/dev/mapper/centos-root  3.8G  3.4G  412M  90% /
devtmpfs                 2.9G     0  2.9G   0% /dev
tmpfs                    2.9G     0  2.9G   0% /dev/shm
tmpfs                    2.9G  9.2M  2.9G   1% /run
tmpfs                    2.9G     0  2.9G   0% /sys/fs/cgroup
/dev/mapper/centos-home  7.5G   65M  7.4G   1% /home
/dev/mapper/centos-var   1.9G  327M  1.6G  18% /var
/dev/sda2                485M  227M  258M  47% /boot
tmpfs                    580M  4.0K  580M   1% /run/user/42
tmpfs                    580M   28K  580M   1% /run/user/1000
/dev/sr0                 8.1G  8.1G     0 100% /run/media/somu/CentOS 7 x86_64
tmpfs                    580M     0  580M   0% /run/user/0
# du -hs /var/log
13M	/var/log
```

Chapter 14

Managing Partitions

14.1 Understanding Disk Layout

There are two basic ways of organizing data on a hard disk : Partitions and LVM (Logical Volume Management). Some parts of a hard disk need to be configured with a fixed amount of storage. In such cases we use partitions. This is applicable for `/boot` and `/` in the figure. However, certain directories contain dynamic user data, and thus need to be able to grow to any size. In such cases, the partitions don't work and we need to use Logical Volumes. In the image below, `sda1, sda2 & sda3` are all Physical Volume(PV)s or partitions. In linux, each partition needs to be connected to one or more directories in order to be used.

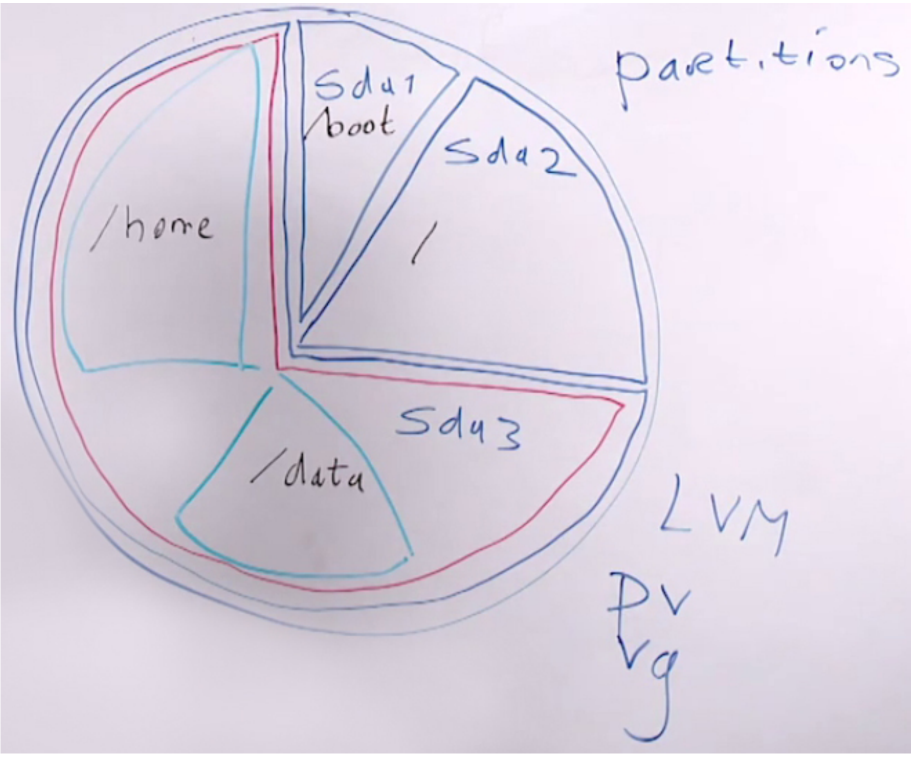

Figure 14.1: Disk Layout

In the case of Logical Volumes (LVs), just like partitions, there needs to be a Physical Volume (PV). This PV is then put in a Volume Group (Vg), represented by the red boundary lines in the image above. From this volume group, we can create Logical Volumes (represented by the blue lines). The advantage of this method is the unused space between different LVs can be added to any of the LVs, and thus no disk space is wasted and no directory in the LV is going to be full while another is barely filled. In LVM it's very easy to grow a logical volume later!

14.2 Creating Partitions

To add a new disk to our OS, first we need to verify the storage disks that are available. For this we use the **proc** filesystem in `/proc/`. It acts as an interface to everything that's happening in the kernel. The `/proc/partitions` file contains a listing of all the disks and partitions that are currently existing.

```
$ cat /proc/partitions
major minor  #blocks  name

8        0   20971520 sda
8        1       2048 sda1
8        2     499712 sda2
8        3   15634432 sda3
8       16   10485760 sdb
11        0    8491008 sr0
253        0    3903488 dm-0
253        1    1953792 dm-1
253        2    1953792 dm-2
253        3    7815168 dm-3
```

While *sda* is the first hard disk, the device *sdb* is a newly added one - the second hard disk avaiable in the computer. The partitions are marked by a number after the device name - *sda1, sda2 and sda3*. The second hard disk doesn't have any partitions yet.

14.2.1 fdisk

fdisk is an old partitioning tool that has been revised for RHEL 7. Running the `fdisk` utility on `/dev/sdb`, the location which the OS uses to designate the second hard disk yeilds:

```
# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table
Building a new DOS disklabel with disk identifier 0xf11ab429.

Command (m for help):
```

It tells us that the disk doesn't contain any partitions (since it's brand new). The fdisk utility offers us a bunch of commands, among which we'll use:

Options	Description
p	Print partition table
n	Add a new partition
w	Write the table to disk and exit

p command

The p option gives us the current disk layout:

```
Command (m for help): p

Disk /dev/sdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xf11ab429

Device Boot      Start         End      Blocks   Id  System

Command (m for help):
```

The device name is *sdb* and its size is 10.7GB. This gives it 20 Million sectors, since the size of each sector is 512B. Now we add a new partition on the disk:

```
Command (m for help): n
Partition type:
p   primary (0 primary, 0 extended, 4 free)
e   extended
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-20971519, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-20971519, default 20971519): +100M
Partition 1 of type Linux and of size 100 MiB is set

Command (m for help):
```

The default option at the prompt can be selected by simply pressing the enter key. Since there are no physical partitions already available, and since we should always choose the option to add physical partitions whenever possible, we add a new physical partition. It asks us for the starting sector, the default of which is 2048. The first 2MBs are used to store meta-data. Next, we mark the end of the partition using a relative size: in this case, of 100MiB (1024^2B). The size has to be specified with uppercase K/M/G to not be misconstrued to any other unit. Printing the partition table now shows:

```
Device Boot          Start          End       Blocks    Id  System
/dev/sdb1             2048       206847       102400    83  Linux
```

This new partition can then be written to the disk using `w`.

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

Now when we view the partitions in `/proc/partitions` we see:

```
# cat /proc/partitions
major minor  #blocks  name
```

```

8        0   20971520 sda
8        1       2048 sda1
8        2     499712 sda2
8        3   15634432 sda3
8       16   10485760 sdb
8       17     102400 sdb1
11        0    8491008 sr0
253        0    3903488 dm-0
253        1    1953792 dm-1
253        2    1953792 dm-2
253        3    7815168 dm-3
```

The disk now has a sdb1 partition of size 100MiB. This indicates that the disk is now ready to accept a filesystem. In case an error is show along the lines of "*the device is busy*", the system probably needs a restart.

14.3 Understanding File System Differences

For a RHEL 7 installation, there are several file system choices:

File System	Description
XFS	The default file system for RHEL 7 and many others, built with scalability in mind. Based on a B-Tree database, which specializes in disk space allocation with high speed and makes looking up files really easy. It also has different tuning options for varying workloads.
Ext4	This was the default filesystem till RHEL 6. It was based on 1993's Ext2 file system which was built to handle much smaller disks than our current needs. It uses H-Tree indexing, which is use of basic index files - suitable for thousands of files, but not practical or economical (in terms of time) for millions of files, which our systems demand. While it is not as scalable as XFS, it does provide backwards compatibility. Thus, for best performance, it shouldn't be used as a default file system.
Btrfs	This is a Copy-on-Write(CoW) file system, which means that before writing to a file, that file is copied somewhere else, thus making journaling unnecessary! Journalling is the system where the filesystem keeps track of the changes being made to the file to make rolling back possible. This also makes undelete operations unnecessary (which have never worked on Linux anyway). It even has added features like subvolumes. It wasn't however included in RHEL 7 First Customer Shipment (FCS).
vfat	Primarily for compatibility with other OSs, such as Windows. It is particularly useful for removable media such as USB sticks. This filesystem is not needed to be installed on the hard disk of the server however, even in cases where Samba provides access to files on the server (Samba handles the file system differences and translation itself).
GFS2	For Active/Active High Availability (HA) Clustering Environments. Only needed when multiple nodes need to write to the same file system concurrently. For Active/Passive File HA Clusters, XFS/Ext4/etc. suffice.
Gluster	Gluster is a distributed file system. Thus, even though represented under the same hierarchy, it can reside on multiple servers. Storage is configured to be done in *bricks* that are spread over servers. Each brick uses XFS as their back-end file system. This is an important file system for cloud environments.

14.4 Making the File System

Just after being created, a partition contains no file system, and thus no files can yet be stored on it. We have to create an appropriate file system using:

14.4.1 mkfs

This is actually a whole bunch of different utilities that are grouped together under the same command. They are:

```
mkfs         mkfs.btrfs   mkfs.cramfs  mkfs.ext2    mkfs.ext3    mkfs.ext4    mkfs.fat
 ↪  mkfs.minix   mkfs.msdos   mkfs.vfat    mkfs.xfs
```

Since the default file system is XFS, mkfs.xfs is the default file system utility.

```
# mkfs.xfs --help
mkfs.xfs: invalid option -- '-'
unknown option --
Usage: mkfs.xfs
/* blocksize */                [-b log=n|size=num]
/* metadata */                [-m crc=0|1,finobt=0|1,uuid=xxx]
/* data subvol */        [-d agcount=n,agsize=n,file,name=xxx,size=num,
(sunit=value,swidth=value|su=num,sw=num|noalign),
sectlog=n|sectsize=num
/* force overwrite */        [-f]
/* inode size */        [-i log=n|perblock=n|size=num,maxpct=n,attr=0|1|2,
projid32bit=0|1]
/* no discard */        [-K]
/* log subvol */        [-l agnum=n,internal,size=num,logdev=xxx,version=n
sunit=value|su=num,sectlog=n|sectsize=num,
lazy-count=0|1]
/* label */                [-L label (maximum 12 characters)]
/* naming */                [-n log=n|size=num,version=2|ci,ftype=0|1]
/* no-op info only */        [-N]
/* prototype file */        [-p fname]
/* quiet */                [-q]
/* realtime subvol */        [-r extsize=num,size=num,rtdev=xxx]
/* sectorsize */        [-s log=n|size=num]
/* version */                [-V]
devicename
<devicename> is required unless -d name=xxx is given.
<num> is xxx (bytes), xxxs (sectors), xxxb (fs blocks), xxxk (xxx KiB),
xxxm (xxx MiB), xxxg (xxx GiB), xxxt (xxx TiB) or xxxp (xxx PiB).
<value> is xxx (512 byte blocks).
```

The **block size (-b)** should be larger when primarily dealing with large files. This way, lesser blocks are used, and the administration of large files becomes easier. The **inode size (-i)** should be larger if it is known beforehand that lots of advanced stuff that stores metadata in inodes will be used. The *label (-L)* sets the name for that filesystem. To actually create the file system, we use the command:

```
meta-data=/dev/sdb1              isize=512    agcount=4, agsize=65536 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=0, sparse=0
data     =                       bsize=4096   blocks=262144, imaxpct=25
```

```
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

14.5 Mounting the Partition Manually

The new partition is mounted using the `mount` command. For recurring mounting, it's advisable to create a permanent mounting directory. For temporary mounts, we can use `/mnt`. The mounting operation can be verified by typing the `mount` command. The command to mount the new partition `sdb1` on the `/mnt` directory is :

```
# mount /dev/sdb1 /mnt
# mount | tail -n 5
tmpfs on /run/user/1000 type tmpfs
↪  (rw,nosuid,nodev,relatime,seclabel,size=592968k,mode=700,uid=1000,gid=1000)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse
↪  (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
/dev/sr0 on /run/media/somu/CentOS 7 x86_64 type iso9660
↪  (ro,nosuid,nodev,relatime,uid=1000,gid=1000,iocharset=utf8,mode=0400,dmode=0500,uhelper=udisks2)
/dev/sdb1 on /mnt type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

Mounting means connecting some device or functionality to a particular directory. This not only includes removable media and peripheral device directories, but also many system settings (such as the `/proc` file system or the `/sys` file system).

To view only the devices that have been mounted, we can use:

```
# mount | grep ^/dev
/dev/mapper/centos-root on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/mapper/centos-home on /home type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sda2 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/mapper/centos-var on /var type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sr0 on /run/media/somu/CentOS 7 x86_64 type iso9660
↪  (ro,nosuid,nodev,relatime,uid=1000,gid=1000,iocharset=utf8,mode=0400,dmode=0500,uhelper=udisks2)
/dev/sdb1 on /mnt type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

14.5.1 umount

The `umount` command is used to unmount a mounted directory. This is to ensure that no files are open and cannot be damaged by the sudden removal of the file system. The `umount` command takes as parameter either the device name or the directory where it is mounted. So, both `/dev/sdb1` and `/mnt` are valid parameters to unmount the new partition.

```
# umount /dev/sdb1
# mount | grep ^/dev
/dev/mapper/centos-root on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/mapper/centos-home on /home type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sda2 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/mapper/centos-var on /var type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sr0 on /run/media/somu/CentOS 7 x86_64 type iso9660
↪  (ro,nosuid,nodev,relatime,uid=1000,gid=1000,iocharset=utf8,mode=0400,dmode=0500,uhelper=udisks2)
```

The device /dev/sdb1 is no longer mounted, as can be seen from the output. A major challenge that may be presented by this is the fact that the device names may change at any time! Today the device that's called /dev/sdb1 may change to /dev/sdc1 if the OS detects the devices (and names them) in another order, thus making our references to them useless. For this reason the *Universally Unique ID (UUID)* of a device can be used to refer to it. The UUID of all existing devices can be displayed using:

```
# blkid
/dev/sda2: UUID="1c55e935-8099-49c4-8c72-0bc1ff7c396a" TYPE="xfs"
/dev/sda3: UUID="DfepDW-igyh-eI6D-SgBB-3HV5-QTQT-EI3Pc2" TYPE="LVM2_member"
/dev/sdb1: LABEL="myfs" UUID="00a8c244-8781-492c-a6ad-85624780e1e8" TYPE="xfs"
/dev/sr0: UUID="2017-09-06-10-53-42-00" LABEL="CentOS 7 x86_64" TYPE="iso9660" PTTYPE="dos"
/dev/mapper/centos-root: UUID="d2fe3168-4eef-431b-9a8e-eb59dae10bcb" TYPE="xfs"
/dev/mapper/centos-swap: UUID="24b0103c-d574-4623-bc85-9255076e3b7d" TYPE="swap"
/dev/mapper/centos-var: UUID="ed13b5f3-1b26-48f7-81cb-03a2bad5fc61" TYPE="xfs"
/dev/mapper/centos-home: UUID="710a33e6-7e52-4c06-830d-e53ae0d58fed" TYPE="xfs"
```

As can be seen, the label for the file system is also shown using the blkid command. Both the UUID and the label for the file system can be used to reference the device while using the mount command:

```
# mount LABEL=myfs /mnt
# mount | tail -n 3
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
/dev/sr0 on /run/media/somu/CentOS 7 x86_64 type iso9660 (ro,nosuid,nodev,relatime,uid=1000,gid=1000,iocharset=utf8,mode=0400,dmode=0500,uhelper=udisks2)
/dev/sdb1 on /mnt type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

14.6 Understanding /etc/fstab

The names of the devices in /etc/fstab are not based on their actual device names, but either the LVM volume names or their UUID. The typical /etc/fstab file looks like:

```
# cat fstab

#
# /etc/fstab
# Created by anaconda on Sat Nov 25 08:44:04 2017
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/centos-root /                       xfs     defaults        0 0
UUID=1c55e935-8099-49c4-8c72-0bc1ff7c396a /boot                   xfs     defaults        0 0
/dev/mapper/centos-home /home                   xfs     defaults        0 0
/dev/mapper/centos-var  /var                    xfs     defaults        0 0
/dev/mapper/centos-swap swap                    swap    defaults        0 0
```

The first parameter is the UUID or the LVM volume name. The second is the directory in the FHS where the disk will be mounted. This is followed by the file system type and then the mount options. The first among the last two numbers is the option for backup support, specifically an old utility called dump-backup. Some enterprise backup utilities need dump functionality provided by this option to operate, even though they don't really use the dump-backup program. The last number is `fsck` - file system check. The concept is to check the file system at boot time before it is mounted and data on it is accessed. There are three valid options: **0** (off), **1** (check the root [/] file system) and finally **2** (check non-root file system).

14.7 Mounting partitions via /etc/fstab

To automount the device `/dev/sdb1`, all we need to do is add the following line to `/etc/fstab`:

```
/dev/sdb1        /data       xfs                defaults        1        2
```

Note however that the above won't guarantee that the *proper* file system will always be mounted as it's dependent on the order in which the OS detects the disks! So, it's better to use the UUIDs of the file systems to track them. We use the UUID to auto mount the file system with:

```
#
# /etc/fstab
# Created by anaconda on Sat Nov 25 08:44:04 2017
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/centos-root /                       xfs     defaults        0 0
UUID=1c55e935-8099-49c4-8c72-0bc1ff7c396a /boot                   xfs     defaults        0 0
/dev/mapper/centos-home /home                   xfs     defaults        0 0
/dev/mapper/centos-var  /var                    xfs     defaults        0 0
/dev/mapper/centos-swap swap                    swap    defaults        0 0
UUID=00a8c244-8781-492c-a6ad-85624780e1e8 /data xfs     defaults        1 2
```

To confirm the mount, we use the `mount -a` command, which tries to mount everything that hasn't been mounted yet.

```
# mount -a
mount: mount point /data does not exist
```

In this case, since the directory `/data` doesn't exist, the mounting failed. The mount system doesn't create a new directory (mounting location) if it doesn't yet exist. We remedy this by:

```
# ls
bin  boot  dev  downloads  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
# mkdir data
# mount -a
# mount | grep ^/dev/
/dev/mapper/centos-root on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/mapper/centos-home on /home type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sda2 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/mapper/centos-var on /var type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sr0 on /run/media/somu/CentOS 7 x86_64 type iso9660 (ro,nosuid,nodev,relatime,uid=1000,gid=1000,iocharset=utf8,mode=0400,dmode=0500,uhelper=udisks2)
/dev/sdb1 on /mnt type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sdb1 on /data type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

As is evident from the last line of the output, the file system has been properly mounted and is ready for use. An alternative way to mount it using fstab would've been to use the label for the disk instead of the UUID, such as:

```
LABEL=myfs        /data        xfs        defaults        1        2
```

14.7.1 Managing xfs file systems using xfs_ commands

The new xfs file system provides a set of commands that start with `xfx_` that help administer any xfs partition. They are:

```
xfs_admin      xfs_copy       xfs_estimate   xfs_fsr        xfs_info
xfs_logprint   xfs_metadump   xfs_ncheck     xfs_repair     xfs_bmap
xfs_db         xfs_freeze     xfs_growfs     xfs_io         xfs_mdrestore
xfs_mkfile     xfs_quota      xfs_rtcp
```

To add a new label to the boot device, we use `xfs_admin` command with the `-L` command. Let us say, we want to label the boot partition on our system. To find out which partition is mapped to `/boot` we use the `lsblk` command.

```
# lsblk
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda               8:0    0   20G  0 disk
sda1            8:1    0    2M  0 part
sda2            8:2    0  488M  0 part /boot
sda3            8:3    0 14.9G  0 part
centos-root 253:0    0  3.7G  0 lvm  /
centos-swap 253:1    0  1.9G  0 lvm  [SWAP]
centos-var  253:2    0  1.9G  0 lvm  /var
centos-home 253:3    0  7.5G  0 lvm  /home
sdb               8:16   0   10G  0 disk
sdb1            8:17   0    1G  0 part /data
sr0              11:0    1  8.1G  0 rom  /run/media/somu/CentOS 7 x86_64
# blkid
/dev/sda2: UUID="1c55e935-8099-49c4-8c72-0bc1ff7c396a" TYPE="xfs"
/dev/sda3: UUID="DfepDW-igyh-eI6D-SgBB-3HV5-QTQT-EI3Pc2" TYPE="LVM2_member"
/dev/sdb1: LABEL="myfs" UUID="00a8c244-8781-492c-a6ad-85624780e1e8" TYPE="xfs"
/dev/sr0: UUID="2017-09-06-10-53-42-00" LABEL="CentOS 7 x86_64" TYPE="iso9660"
 ↪  PTTYPE="dos"
/dev/mapper/centos-root: UUID="d2fe3168-4eef-431b-9a8e-eb59dae10bcb" TYPE="xfs"
/dev/mapper/centos-swap: UUID="24b0103c-d574-4623-bc85-9255076e3b7d" TYPE="swap"
/dev/mapper/centos-var: UUID="ed13b5f3-1b26-48f7-81cb-03a2bad5fc61" TYPE="xfs"
/dev/mapper/centos-home: UUID="710a33e6-7e52-4c06-830d-e53ae0d58fed" TYPE="xfs"
# xfs_admin -L bootdevice /dev/sda2
xfs_admin: /dev/sda2 contains a mounted filesystem

fatal error -- couldn't initialize XFS library
```

The `/boot` partition was auto-mounted at start up, and thus it needs to be unmounted first before it can be edited. So, we take the following steps:

```
# umount /boot
# xfs_admin -L bootdevice /dev/sda2
writing all SBs
new label = "bootdevice"
```

```
# mount -a
# blkid
/dev/sda2: LABEL="bootdevice" UUID="1c55e935-8099-49c4-8c72-0bc1ff7c396a" TYPE="xfs"
/dev/sda3: UUID="DfepDW-igyh-eI6D-SgBB-3HV5-QTQT-EI3Pc2" TYPE="LVM2_member"
/dev/sdb1: LABEL="myfs" UUID="00a8c244-8781-492c-a6ad-85624780e1e8" TYPE="xfs"
/dev/sr0: UUID="2017-09-06-10-53-42-00" LABEL="CentOS 7 x86_64" TYPE="iso9660" PTTYPE="dos"
/dev/mapper/centos-root: UUID="d2fe3168-4eef-431b-9a8e-eb59dae10bcb" TYPE="xfs"
/dev/mapper/centos-swap: UUID="24b0103c-d574-4623-bc85-9255076e3b7d" TYPE="swap"
/dev/mapper/centos-var: UUID="ed13b5f3-1b26-48f7-81cb-03a2bad5fc61" TYPE="xfs"
/dev/mapper/centos-home: UUID="710a33e6-7e52-4c06-830d-e53ae0d58fed" TYPE="xfs"
```

14.8 Understanding Encrypted Partitions

VIDEO TUTORIAL MISSING

14.9 Creating a LUKS Encrypted Partition

To create the encrypted partition, we once again use the `fdisk` utility. Since we want to put this partition on the `/dev/sdb` device, we use:

```
# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): p

Disk /dev/sdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xf11ab429

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1         2048     2099199     1048576   83  Linux
```

At first we ensure that sufficient disk space is available via printing the existing file system details on the disk. In this example, we can see that the number of available sectors on the disk is $20,971,520$ while the End=$2,099,199$ tells us that only those sectors are used till now. Thus, we can add a new encrypted partition. The initial procedure is same as creating a normal partition:

```
Command (m for help): n
Partition type:
p   primary (1 primary, 0 extended, 3 free)
e   extended
Select (default p): p
Partition number (2-4, default 2):
First sector (2099200-20971519, default 2099200):
```

```
Using default value 2099200
Last sector, +sectors or +size{K,M,G} (2099200-20971519, default 20971519): +100M
Partition 2 of type Linux and of size 100 MiB is set

Command (m for help): p

Disk /dev/sdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xf11ab429

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1         2048     2099199     1048576   83  Linux
/dev/sdb2      2099200     2303999      102400   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.

# partprobe /dev/sdb
# cat /proc/partitions
major minor  #blocks  name

8        0   20971520 sda
8        1       2048 sda1
8        2     499712 sda2
8        3   15634432 sda3
8       16   10485760 sdb
8       17    1048576 sdb1
8       18     102400 sdb2
11       0    8491008 sr0
253      0    3903488 dm-0
253      1    1953792 dm-1
253      2    1953792 dm-2
253      3    7815168 dm-3
```

The `partprobe /dev/sdb` command updates the kernel parititon table, i.e., tells the kernel about the changes in the partition table on that device so that the kernel can provide the required functionality.

14.9.1 Formatting the new partition

To encrypt the new partition we use the `cryptsetup` command. An argument of `luksFormat` is required to specify the encryption formatting. We encrypt `/dev/sdb2` by:

```
# cryptsetup luksFormat /dev/sdb2

WARNING!
========
This will overwrite data on /dev/sdb2 irrevocably.
```

```

Are you sure? (Type uppercase yes): YES
Enter passphrase:
Verify passphrase:
#
```

Now that our encrypted partition has been created, we need to open it to use it. For this we can use a custom-made dedicated mount point such as `/secret`. Then, we have to open the partition using `cryptsetup luskOpen` before we can mount it. At this point, we also have to provide a name for the partition. Finally, we go to the `/dev/mapper` directory to ensure that the new partition has been successfully loaded.

```
# cryptsetup luksOpen /dev/sdb2 secret
Enter passphrase for /dev/sdb2:
[root@vmPrime /]# cd /dev/mapper
[root@vmPrime mapper]# ls
centos-home  centos-root  centos-swap  centos-var  control  secret
```

Since we can see the required partition in the `/dev/mapper` directory, we can be sure that the partition was opened successfully! The complete path of our partition is `/dev/mapper/secret`. Now we can proceed to create a file system on it: (let's assume we want to format the disk as ext4)

```
# mkfs.ext4 /dev/mapper/secret
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
25168 inodes, 100352 blocks
5017 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=33685504
13 block groups
8192 blocks per group, 8192 fragments per group
1936 inodes per group
Superblock backups stored on blocks:
8193, 24577, 40961, 57345, 73729

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

The encrypted partition is now ready to be mounted and used. We do this by:

```
# mkdir /secret
# mount /dev/mapper/secret /secret
# cd /secret
```

While we normally might never need to close the encrypted partition, if for example we have a partition that's stored on an external device, we first need to unmount it, followed by performing a `cryptsetup luksClose`.

```
[root@vmPrime ~]# umount /secret
[root@vmPrime ~]# cryptsetup luksClose /dev/mapper/secret
```

```
[root@vmPrime ~]# ls -l /dev/mapper
total 0
lrwxrwxrwx. 1 root root       7 Dec  8 06:51 centos-home -> ../dm-3
lrwxrwxrwx. 1 root root       7 Dec  8 06:51 centos-root -> ../dm-0
lrwxrwxrwx. 1 root root       7 Dec  8 06:51 centos-swap -> ../dm-1
lrwxrwxrwx. 1 root root       7 Dec  8 06:51 centos-var -> ../dm-2
crw-------. 1 root root 10, 236 Dec  8 06:51 control
```

The folder `/secret` within `/dev/mapper` has disappeared as it's been saved to the original encrypted partition where no one can access the data without decryption.

Now, if we want to automount the partition, we need to add an entry for it in the `/etc/fstab` file:

```
/dev/mapper/secret        /secret        ext4        defaults        1 2
```

However, the above code work since at the time the `/etc/fstab` file is processes, there is no `/dev/mapper/secret` directory during boot since the file system on the encrypted partition won't be open yet! To do this, we need to create/edit the `/etc/crypttab` file, with the following contents:

```
secret        /dev/sda2        none
```

The first value in the file is the name that's to be assigned to the partition in the `/dev/mapper/` directory, the second is the device name, and the third, the name of password file to be used. Since we're not using a password file, we've left it as *none*. Thus, the system will prompt for the passphrase at boot to open and mount the LUKS encrypted device. Now, to confirm the auto mounting of the device, we need to reboot.

14.10 Dealing with "Enter root password for maintenance mode"

If there is an error within our `/etc/fstab` file, our OS will fail to boot. This puts us in emergency mode, which lets us log in as the root user to troubleshoot. Since it's a boot-time error, a good idea is to use the `journalctl -xb` to view the `journald` logs, which may help us locate the problem.

If the error is along the line of "the device timed out", then there is probably a typo in the device name. We can then fix the error and reboot the server.

Chapter 15

Managing LVM Logical Volumes

15.1 Why use LVM

LVM provides a flexible approach to storage:

- Volumes can consist of more than one disk.
- Volumes can be made smaller or larger easily.
- It is easy to replace failing disks.
- Provides advanced options like working with snapshots - a method by which backups can be made of files while they're open!
- It is easy to add many new volumes. While with partitions, there is a limit of 15 partitions, there can be as many as 256 logical volumes.

15.2 Understanding LVM Setup

When working with LVM, we always start with physical storage media such as a hard disk (*sda*). Typically, a partition on the hard disk is marked as the physical volume. Now, this physical volume is added to the **volume group** which is essentially an abstraction of all the storage available. Thus, all logical volumes are created from this volume group, and from their perspective, the physical volume that's acting as it's storage media isn't important.

Once the logical volume is made from the storage in the volume group, we get a device called `/dev/vgroup/logvol`. This is the device for the logical volume on which we create the file system. As long as there is space on the volume group, we can add new logical volumes on it. If there isn't we can also add a physical volume to the volume group, to increase its capacity.

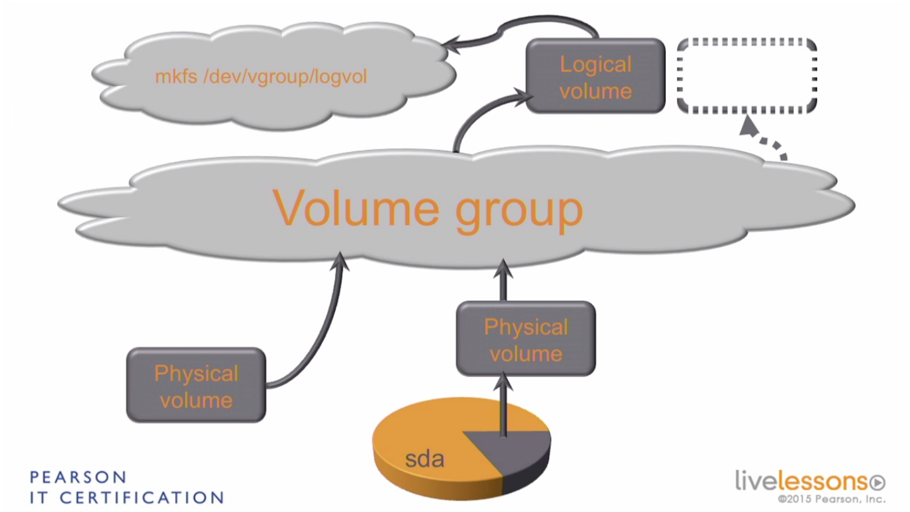

Figure 15.1: LVM Setup

15.3 Creating an LVM Logical Volume

To create a new LVM partition, first we need to make a partition like any other partition.

```
# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): p

Disk /dev/sdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xf11ab429

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048     4196351     2097152    5  Extended
/dev/sdb5            4096     2101247     1048576   83  Linux
/dev/sdb6         2103296     4196351     1046528   83  Linux

Command (m for help): n
Partition type:
p   primary (0 primary, 1 extended, 3 free)
l   logical (numbered from 5)
Select (default p): p
Partition number (2-4, default 2):
First sector (4196352-20971519, default 4196352):
Using default value 4196352
Last sector, +sectors or +size{K,M,G} (4196352-20971519, default 20971519): +100M
Partition 2 of type Linux and of size 100 MiB is set

Command (m for help): p

```

```
Disk /dev/sdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xf11ab429

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048     4196351     2097152    5  Extended
/dev/sdb2         4196352     4401151      102400   83  Linux
/dev/sdb5            4096     2101247     1048576   83  Linux
/dev/sdb6         2103296     4196351     1046528   83  Linux
# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): p

Disk /dev/sdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xf11ab429

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048     4196351     2097152    5  Extended
/dev/sdb5            4096     2101247     1048576   83  Linux
/dev/sdb6         2103296     4196351     1046528   83  Linux

Command (m for help): n
Partition type:
p   primary (0 primary, 1 extended, 3 free)
l   logical (numbered from 5)
Select (default p): p
Partition number (2-4, default 2):
First sector (4196352-20971519, default 4196352):
Using default value 4196352
Last sector, +sectors or +size{K,M,G} (4196352-20971519, default 20971519): +100M
Partition 2 of type Linux and of size 100 MiB is set

Command (m for help): p

Disk /dev/sdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xf11ab429

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048     4196351     2097152    5  Extended
/dev/sdb2         4196352     4401151      102400   83  Linux
/dev/sdb5            4096     2101247     1048576   83  Linux
/dev/sdb6         2103296     4196351     1046528   83  Linux
```

Now that we have specified the details of our new partition, we need to change one more before we can use the LVM partitions. We enter the command `t` to change the partition type, and then show the overview of all the acceptable partition types using `l`. From the command below, we can see that there is a partition type called **Linux LVM** which suits our requirements.

```
Command (m for help): t
Partition number (1,2,5,6, default 6): 2
Hex code (type L to list all codes): L

0  Empty           24  NEC DOS         81  Minix / old Lin bf  Solaris
1  FAT12           27  Hidden NTFS Win 82  Linux swap / So c1  DRDOS/sec (FAT-
2  XENIX root      39  Plan 9          83  Linux           c4  DRDOS/sec (FAT-
3  XENIX usr       3c  PartitionMagic  84  OS/2 hidden C:  c6  DRDOS/sec (FAT-
4  FAT16 <32M      40  Venix 80286     85  Linux extended  c7  Syrinx
5  Extended        41  PPC PReP Boot   86  NTFS volume set da  Non-FS data
6  FAT16           42  SFS             87  NTFS volume set db  CP/M / CTOS / .
7  HPFS/NTFS/exFAT 4d  QNX4.x          88  Linux plaintext de  Dell Utility
8  AIX             4e  QNX4.x 2nd part 8e  Linux LVM       df  BootIt
9  AIX bootable    4f  QNX4.x 3rd part 93  Amoeba          e1  DOS access
a  OS/2 Boot Manag 50  OnTrack DM      94  Amoeba BBT      e3  DOS R/O
b  W95 FAT32       51  OnTrack DM6 Aux 9f  BSD/OS          e4  SpeedStor
c  W95 FAT32 (LBA) 52  CP/M            a0  IBM Thinkpad hi eb  BeOS fs
e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a5  FreeBSD         ee  GPT
f  W95 Ext'd (LBA) 54  OnTrackDM6      a6  OpenBSD         ef  EFI (FAT-12/16/
10  OPUS            55  EZ-Drive        a7  NeXTSTEP        f0  Linux/PA-RISC b
11  Hidden FAT12    56  Golden Bow      a8  Darwin UFS      f1  SpeedStor
12  Compaq diagnost 5c  Priam Edisk     a9  NetBSD          f4  SpeedStor
14  Hidden FAT16 <3 61  SpeedStor       ab  Darwin boot     f2  DOS secondary
16  Hidden FAT16    63  GNU HURD or Sys af  HFS / HFS+      fb  VMware VMFS
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs         fc  VMware VMKCORE
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap       fd  Linux raid auto
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid fe  LANstep
1c  Hidden W95 FAT3 75  PC/IX           be  Solaris boot    ff  BBT
1e  Hidden W95 FAT1 80  Old Minix
Hex code (type L to list all codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'

Command (m for help): p

Disk /dev/sdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xf11ab429

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048     4196351     2097152    5  Extended
/dev/sdb2         4196352     4401151      102400   8e  Linux LVM
/dev/sdb5            4096     2101247     1048576   83  Linux
/dev/sdb6         2103296     4196351     1046528   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
# partprobe
```

We enter the value `8e` since it's the code for the Linux LVM partition that we need. Finally, we use `p` to print the partition table and verify our partition, `w` to save the changes and `partprobe` to push the changes to the kernel.

15.3.1 Creating a Physical Volume

A physical volume is just a partition with the LVM metadata added to it. The volume groups built from the PVs are not possible to build without this metadata stored in the partitions. The physical volumes are created using `pvcreate`. We can show all physical volumes using `pvs`.

```
# pvcreate /dev/sdb2
Physical volume "/dev/sdb2" successfully created.
# pvs
PV         VG     Fmt  Attr PSize   PFree
/dev/sda3  centos lvm2 a--  <14.91g   4.00m
/dev/sdb2         lvm2 ---  100.00m 100.00m
```

The `pvs` command tells us that we have a physical volume called `/dev/sdb2` which isn't in a volume group yet, has a LVM2 formatting, has a partition size of 100MB and has the same amount of free space.

15.3.2 Creating a Volume Group

Next, we create a new volume group using the `vgcreate` command. Again, we can check the volume groups on our system using the `vgs` command.

```
# vgcreate vgPrime /dev/sdb2
Volume group "vgPrime" successfully created
# vgs
VG      #PV #LV #SN Attr   VSize   VFree
centos    1   4   0 wz--n- <14.91g  4.00m
vgPrime   1   0   0 wz--n-  96.00m 96.00m
```

The volume group *vgPrime* has 1 PV in it (*/dev/sda2*), no logical volumes, And has a Volume size of 96MB, all of which is free!

15.3.3 Creating a Logical Volume

The creating of a Logical Volume on a VG requires specifying the size of the logical volume. This can be done in two ways: by counting the number of extents (building blocks of LVM) [`-l`] or the actual size on disk (KB, MB, GB, TB, etc.)[`-L`]. Finally, we also can provide the name of the LV using the `-n` option.

```
# lvcreate -n lvPrime -L 96M vgPrime
Logical volume "lvPrime" created.
# lvs
LV      VG      Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
home    centos  -wi-ao----  7.45g
root    centos  -wi-ao----  3.72g
```

```
swap    centos  -wi-ao----  1.86g
var     centos  -wi-ao----  1.86g
lvPrime vgPrime -wi-a----- 96.00m
```

15.3.4 Creating a File system on the LV

Now, since the LV is ready, we can put a file system on it. We refer to the logical volume device by /dev/<volumeGroupName>/<logicalVolumeName>.

```
# mkfs.ext2 /dev/vgPrime/lvPrime
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
24576 inodes, 98304 blocks
4915 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67371008
12 block groups
8192 blocks per group, 8192 fragments per group
2048 inodes per group
Superblock backups stored on blocks:
8193, 24577, 40961, 57345, 73729

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done

# mount /dev/vgPrime/lvPrime /mnt
# mount | grep ^/dev/
/dev/mapper/centos-root on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sdb5 on /data type ext4 (rw,relatime,seclabel,data=ordered)
/dev/mapper/centos-var on /var type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/mapper/centos-home on /home type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sda2 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sr0 on /run/media/somu/CentOS 7 x86_64 type iso9660
↪ (ro,nosuid,nodev,relatime,uid=1000,gid=1000,iocharset=utf8,mode=0400,dmode=0500,uhelper=udisks2)
/dev/mapper/vgPrime-lvPrime on /mnt type ext2 (rw,relatime,seclabel)
```

Here, we can see that there is a strange behavior with the device that is mounted. While we issued the command to mount `/dev/vgPrime/lvPrime`, the device that was actually mounted is shown as `/dev/mapper/vgPrime-lvPrime`. This is because both those names are symlinks to the real name of the device (*dm-5*):

```
# ls -l /dev/mapper/vgPrime-lvPrime /dev/vgPrime/lvPrime
lrwxrwxrwx. 1 root root 7 Dec 11 15:13 /dev/mapper/vgPrime-lvPrime -> ../dm-5
lrwxrwxrwx. 1 root root 7 Dec 11 15:13 /dev/vgPrime/lvPrime -> ../dm-5
```

The device `/dev/dm-5` is a Device Mapper device, which is the same as used in case of LUKS encrypted volumes.

15.4 Understanding Device Mapper and LVM Device Names

The device mapper is an abstraction layer that the kernel works with to communicate with certain types of storage devices. Both LUKS encrypted partitions and LVM use the device mapper. Other devices such as software RAID and multipath also have to communicate via the device mapper.

Contrastingly, the XFS, Ext4, etc file systems work with the help of the VFS (Virtual File System) layer (instead of the Device Mapper abstraction layer). The device mapper has the devices present as `dm-*` (*dm-0, dm-1, etc.*) but we shouldn't use them. The names are assigned during boot, and are subject to change at any time! This is why the device mapper provides a bunch of symlinks to the related devices in the `/device/mapper` directory. They are: `/dev/mapper/vg-lv` and `/dev/vg/lv` which are both symlinks to the same device.

15.5 Understanding LVM resize operations

The structure of LVMs are simple: the file system (FS) are installed on Logical Volumes (LV). These Logical Volumes get their disk space from Volume Groups (VG) which use several Physical Volumes (PV) that actually hold the data and provides the disk space.

15.5.1 Extending the File System

To expand the disk capacity of the file system, we need more space in the Logical volume. This means that (possibly) more space has to be added to the Volume Group itself, and thus, more physical volumes may need to be added.

So, first we need to create a new physical volume, and then assign it to the volume group. Then it is possible to grow the logical volume, and finally extend the file system.

15.5.2 Shrinking the File System

At first we have to reduce the size of the file system, and then reduce the size of the logical volume. If we don't there will be a file system with a bigger size than the logical partition it's residing on.

Thus, after reducing the file system size and then the logical volume size, we can then reduce the size of the volume group (if needed).

15.6 Growing an LVM Logical Volume

We can grow the LVM volume if we're running out of disk space and want to make it bigger. We typically check the amount of free space using `df -h` (*disk free - human-readable*) command:

```
# df -h
Filesystem                    Size  Used Avail Use% Mounted on
/dev/mapper/centos-root       3.8G  3.6G  163M  96% /
devtmpfs                      2.9G     0  2.9G   0% /dev
tmpfs                         2.9G     0  2.9G   0% /dev/shm
```

```
tmpfs                        2.9G  9.1M  2.9G   1% /run
tmpfs                        2.9G     0  2.9G   0% /sys/fs/cgroup
/dev/sdb5                    976M  2.6M  907M   1% /data
/dev/mapper/centos-var       1.9G  365M  1.5G  20% /var
/dev/mapper/centos-home      7.5G   68M  7.4G   1% /home
/dev/sda2                    485M  266M  220M  55% /boot
tmpfs                        580M  4.0K  580M   1% /run/user/42
tmpfs                        580M   36K  580M   1% /run/user/1000
/dev/sr0                     8.1G  8.1G     0 100% /run/media/somu/CentOS 7 x86_64
/dev/mapper/vgPrime-lvPrime   93M  1.6M   87M   2% /mnt
```

Now, since it's an LVM, the order in which we grow the different components matter. To make the filesystem bigger, first we check the LV size and then check to see if there's any free space in the VG:

```
# lvs
LV    VG               Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync
 ↪  Convert
root  centos_cliserver -wi-ao---- <17.00g
swap  centos_cliserver -wi-ao----   2.00g
lvCLI vgCLI            -wi-a----- 100.00m
# vgs
VG               #PV #LV #SN Attr   VSize   VFree
centos_cliserver   1   2   0 wz--n- <19.00g      0
vgCLI              2   1   0 wz--n- 192.00m  92.00m
```

From the last line of the command, we can see that vgPrime has 0 VFree (i.e., free space in the VG). Thus, we need to work bottom up and make it bigger before we can make the LV and the FS bigger. So, we run `fdisk` on */dev/sdb*.

```
# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): p

Disk /dev/sdb: 4294 MB, 4294967296 bytes, 8388608 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x9287c46d

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048      206847      102400   83  Linux
/dev/sdb2          206848      411647      102400   83  Linux
/dev/sdb3          411648      821247      204800   83  Linux LVM
```

15.6.1 Creating a new logical volume in an extended partition to add to the VG

Now, we add a new partition. However, since on the given disk there's already 3 primary partitions, and there can only be a total of 4 partitions on a disk (max of 3 primary and 1 extended that can contain several logical partitions), the system defaults the last partition to be an extended one.

```
Command (m for help): n
Partition type:
p   primary (3 primary, 0 extended, 1 free)
e   extended
Select (default e):
Using default response e
Selected partition 4
First sector (821248-8388607, default 821248):
Using default value 821248
Last sector, +sectors or +size{K,M,G} (821248-8388607, default 8388607):
Using default value 8388607
Partition 4 of type Extended and of size 3.6 GiB is set

Command (m for help): p

Disk /dev/sdb: 4294 MB, 4294967296 bytes, 8388608 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x9287c46d

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1           2048      206847      102400   83  Linux
/dev/sdb2         206848      411647      102400   83  Linux
/dev/sdb3         411648      821247      204800   83  Linux LVM
/dev/sdb4         821248     8388607     3783680    5  Extended
```

Typically, we want the extended partition to take whatever disk space is left, since otherwise the space is wasted and rendered unusable due to the MBR convention used by BIOS. However, if UEFI is used, the usage of GUID (Globally Unique ID) Partition Tables (GPT) which lifts this restriction.

Now, we have to add a logical partition on the disk. Since all 4 partitions are in use, the system defaults to adding a new logical partition on the extended partition automatically. We add the new partition and then change the partition type (using `t`) to *Linux LVM* by providing the code `8e`.

```
Command (m for help): n
All primary partitions are in use
Adding logical partition 5
First sector (823296-8388607, default 823296):
Using default value 823296
Last sector, +sectors or +size{K,M,G} (823296-8388607, default 8388607): +100M
Partition 5 of type Linux and of size 100 MiB is set

Command (m for help): t
Partition number (1-5, default 5):
Hex code (type L to list all codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'

Command (m for help): p

Disk /dev/sdb: 4294 MB, 4294967296 bytes, 8388608 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
Disk label type: dos
Disk identifier: 0x9287c46d

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048      206847      102400   83  Linux
/dev/sdb2          206848      411647      102400   83  Linux
/dev/sdb3          411648      821247      204800   83  Linux LVM
/dev/sdb4          821248     8388607     3783680    5  Extended
/dev/sdb5          823296     1028095      102400   8e  Linux LVM
```

Now we save the configuration and use the `partprobe` command to update the kernel's information about the available partitions.

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
# partprobe
```

15.6.2 Extending the Volume Group

Next let us consider we want to extend the existing LVM partition on */dev/sdb3*. Then, we use the `vgextend` command to extend the LVM partition. It requires a Volume Group name, and a Physical device path, with the intention of adding the entire physical device to the VG. This is a *shortcut* since we don't have to create a PV on the device to be added (*/dev/sdb5*), as when all conditions are met, the `vgextend` command itself creates a PV on the disk, after which the disk is extended.

```
# vgextend vgCLI /dev/sdb5
Physical volume "/dev/sdb5" successfully created.
Volume group "vgCLI" successfully extended
```

Now, we can extend the logical volume to take up as much space on the VG as we want. We can confirm that our VG has been extended with the `vgs` command, and we can see which PVs are included in it (and confirm if */dev/sdb5* is present in it), using the `pvs` command.

```
# vgs
VG                #PV #LV #SN Attr   VSize   VFree
centos_cliserver    1   2   0 wz--n- <19.00g      0
vgCLI               2   1   0 wz--n- 292.00m 192.00m
# pvs
PV         VG               Fmt  Attr PSize   PFree
/dev/sda2  centos_cliserver lvm2 a--  <19.00g      0
/dev/sdb2                   lvm2 ---  100.00m 100.00m
/dev/sdb3  vgCLI            lvm2 a--  196.00m  96.00m
/dev/sdb5  vgCLI            lvm2 a--   96.00m  96.00m
```

15.6.3 Extending the LV and the File System

The LV is extended using the `lvextend` command, that takes as an argument:

Options	Description
-L	Absolute size in KiB/MiB/GiB
-l	The number of logical extents **OR** a percentage of either the VG size, the LV/PV size or the free space available in the VG, etc.
-r	Also resizes the file system on the LV, irrespective of file system.

The complete `lvextend` command then looks like:

```
# lvextend -l +100%FREE -r /dev/vgCLI/lvCLI
Phase 1 - find and verify superblock...
Phase 2 - using internal log
- zero log...
- scan filesystem freespace and inode maps...
- found root inode chunk
Phase 3 - for each AG...
- scan (but don't clear) agi unlinked lists...
- process known inodes and perform inode discovery...
- agno = 0
- agno = 1
- agno = 2
- agno = 3
- process newly discovered inodes...
Phase 4 - check for duplicate blocks...
- setting up duplicate extent list...
- check for inodes claiming duplicate blocks...
- agno = 0
- agno = 1
- agno = 2
- agno = 3
No modify flag set, skipping phase 5
Phase 6 - check inode connectivity...
- traversing filesystem ...
- traversal finished ...
- moving disconnected inodes to lost+found ...
Phase 7 - verify link counts...
No modify flag set, skipping filesystem flush and exiting.
Size of logical volume vgCLI/lvCLI changed from 100.00 MiB (25 extents) to 292.00 MiB (73 extents).
Logical volume vgCLI/lvCLI successfully resized.
meta-data=/dev/mapper/vgCLI-lvCLI isize=512    agcount=4, agsize=6400 blks
=                        sectsz=512   attr=2, projid32bit=1
=                        crc=1        finobt=0 spinodes=0
data     =                       bsize=4096   blocks=25600, imaxpct=25
=                        sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal               bsize=4096   blocks=855, version=2
=                        sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 25600 to 74752
```

The last few lines are the output from the `mkfs.xfs` command which is used to resize the file system on the disk. Had the filesystem been XFS, the `resize2fs` utility would've been used instead. The result of the operation can be verified using the `df -h` command and checking the file system size.

```
# df -h /dev/vgCLI/lvCLI
Filesystem               Size  Used Avail Use% Mounted on
/dev/mapper/vgCLI-lvCLI  289M   16M  274M   6% /LVM
```

15.7 Shrinking an LVM logical Volume

The shrinking operation of an LVM needs to be supported by the file system on board the LV. This is not the case for XFS as it doesn't support shrinking. **To shrink a LV, the file system on it must be unmounted first!** The size of the FS then must be reduced before shrinking the LV. To resize the Ext4 FS, we use `resize2fs` utility, which is the *xt2/Ext3/Ext4 File System Resizer*.

If we directly try to run the resize2fs on the disk, we'll be advised to run `e2fsck` utility to check file system consistency, i.e., if the file system has any problems with it. So, the commands to reduce the LV are:

```
# e2fsck -f /dev/mapper/vgCLI-lvCLI
e2fsck 1.42.9 (28-Dec-2013)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
lvCLI: 11/25688 files (9.1% non-contiguous), 8896/102400 blocks
# resize2fs /dev/mapper/vgCLI-lvCLI 50M
resize2fs 1.42.9 (28-Dec-2013)
Resizing the filesystem on /dev/mapper/vgCLI-lvCLI to 51200 (1k) blocks.
The filesystem on /dev/mapper/vgCLI-lvCLI is now 51200 blocks long.
# lvreduce -L 50M /dev/mapper/vgCLI-lvCLI
Rounding size to boundary between physical extents: 52.00 MiB.
WARNING: Reducing active logical volume to 52.00 MiB.
THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce vgCLI/lvCLI? [y/n]: y
Size of logical volume vgCLI/lvCLI changed from 100.00 MiB (25 extents) to 52.00 MiB (13
↪  extents).
Logical volume vgCLI/lvCLI successfully resized.
```

Now, if there weren't any errors, we should be able to mount the file system on board the LV.

```
# mount /dev/mapper/vgCLI-lvCLI /LVM/
# mount | grep ^/dev
/dev/mapper/centos_cliserver-root on / type xfs
↪  (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sda1 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/mapper/vgCLI-lvCLI on /LVM type ext4 (rw,relatime,seclabel,data=ordered)
# df -h /dev/vgCLI/lvCLI
Filesystem               Size  Used Avail Use% Mounted on
/dev/mapper/vgCLI-lvCLI   45M  1.1M   40M   3% /LVM
```

In the last line we see that the file system size has been properly reduced.

15.7.1 Reduce both File system and LV in a single step

It is possible to shrink the LV and the on-board FS in a single command: (*The* `-r` *option automatically resizes the FS before shrinking the LV*).

```
# umount /LVM
# lvreduce -L 35M -r /dev/vgCLI/lvCLI
```

```
Rounding size to boundary between physical extents: 36.00 MiB.
fsck from util-linux 2.23.2
lvCLI: 11/13832 files (18.2% non-contiguous), 6886/51200 blocks
resize2fs 1.42.9 (28-Dec-2013)
Resizing the filesystem on /dev/mapper/vgCLI-lvCLI to 36864 (1k) blocks.
The filesystem on /dev/mapper/vgCLI-lvCLI is now 36864 blocks long.

Size of logical volume vgCLI/lvCLI changed from 52.00 MiB (13 extents) to 36.00 MiB (9 extents).
Logical volume vgCLI/lvCLI successfully resized.
# mount /dev/mapper/vgCLI-lvCLI /LVM
# mount | grep ^/dev
/dev/mapper/centos_cliserver-root on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sda1 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/mapper/vgCLI-lvCLI on /LVM type ext4 (rw,relatime,seclabel,data=ordered)
# df -h /dev/mapper/vgCLI-lvCLI
Filesystem               Size  Used Avail Use% Mounted on
/dev/mapper/vgCLI-lvCLI   31M  783K   28M   3% /LVM
```

Note however, that this method won't work all the time on all file systems, due to the fact that the target FS must also support reduction via `lvreduce -r`. Thus, while the `-r` won't work on XFS, it works just fine on Ext4.

Part III

Performing Advanced System Administration Tasks

Chapter 16

Managing the Kernel

16.1 Understanding the Modular Structure of the Kernel

The primary responsibility of the Linux Kernel is addressing the hardware and managing it. By default the kernel contains every functionality required to address the available hardware.

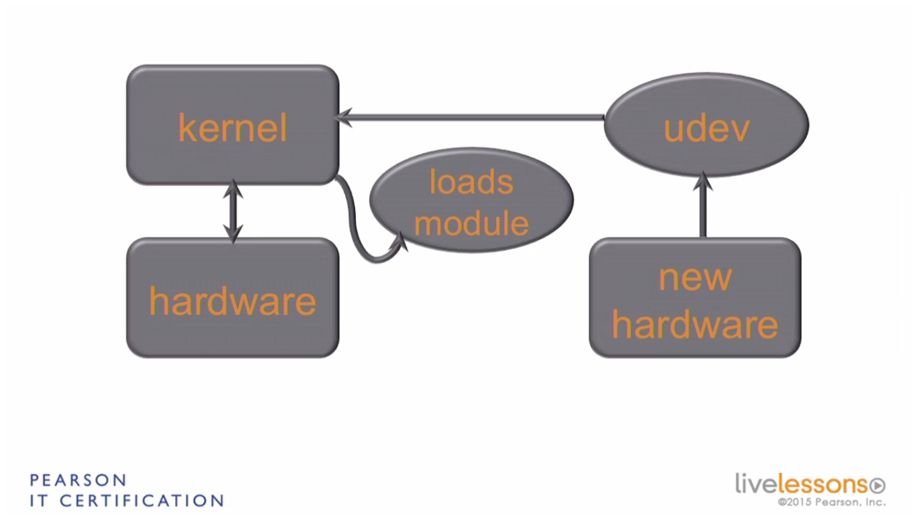

Figure 16.1: Modular structure of the Kernel

When a new hardware is added, the **udev** process is awoken to initialize the new hardware by communicating with the kernel and providing it with all the necessary information about the device. In turn, the kernel will load the module specific to that new hardware and initialize it so that the user can start using it.

16.2 Working with Kernel Modules

The linux kernel works with modules to only load those modules which provided a required functionality. Thus, these kernels are very lean since only the functionalities being used are kept.

16.2.1 Viewling loaded Kernel Modules

The `lsmod` command shows us all the kernel module that have been loaded.

```
# lsmod
Module                  Size  Used by
nls_utf8               12557  1
isofs                  39844  1
fuse                   91874  3
xt_CHECKSUM            12549  1
ipt_MASQUERADE         12678  3
...
dm_region_hash         20813  1 dm_mirror
dm_log                 18411  2 dm_region_hash,dm_mirror
dm_mod                123303  14 dm_log,dm_mirror
```

The provided data is the name and size of the module followed by the number of programs currently using the module. In older versions of linux it was required to manually load modules to access certain functionalities. Modern linux distros don't require this. The **udev** process takes care of loading modules automatically. The activity of this process can be monitored using the `udevadm monitor` command.

```
monitor will print the received events for:
UDEV - the event which udev sends out after rule processing
KERNEL - the kernel uevent
```

Now if we were to attach a pen drive (and then remove it) then that'd trigger the following:

```
1  KERNEL[15445.778458] add      /devices/pci0000:00/0000:00:11.0/0000:02:03.0/usb1/1-1
    ↪  (usb)
2  KERNEL[15445.786957] add
    ↪  /devices/pci0000:00/0000:00:11.0/0000:02:03.0/usb1/1-1/1-1:1.0 (usb)
3  UDEV  [15445.830980] add      /devices/pci0000:00/0000:00:11.0/0000:02:03.0/usb1/1-1
    ↪  (usb)
4  KERNEL[15445.851667] add      /module/usb_storage (module)
5  UDEV  [15445.877116] add      /module/usb_storage (module)
6  ...
7
8  KERNEL[15893.875998] add      /module/fat (module)
9  KERNEL[15893.876022] add      /kernel/slab/fat_cache (slab)
10 KERNEL[15893.876030] add      /kernel/slab/fat_inode_cache (slab)
11 KERNEL[15893.876037] add      /module/vfat (module)
12 UDEV  [15893.878942] add      /module/fat (module)
13 UDEV  [15893.878962] add      /kernel/slab/fat_cache (slab)
14 UDEV  [15893.878969] add      /kernel/slab/fat_inode_cache (slab)
15 UDEV  [15893.879002] add      /module/vfat (module)
16
17 ...
18 UDEV  [15462.218411] remove   /devices/pci0000:00/0000:00:11.0/0000:02:03.0/usb1/1-1
    ↪  (usb)
19 KERNEL[15462.232117] remove   /host3/target3:0:0 (scsi)
20 UDEV  [15462.232957] remove   /host3/target3:0:0 (scsi)
```

The moment new hardware is detected, it generates an event which is then received by udev to respond appropriately. The insertion of the pen drive (usb key) into the usb slot triggered the udev process to load the *fat* and *vfat* modules [Lines `8-15`].

However, when the USB key is detached, the modules aren't unloaded. This can be verified by the presence of both *fat & vfat* modules in the output of:

```
# lsmod | grep fat
vfat                   17461  0
fat                    65950  1 vfat
```

16.2.2 Modprobe

The `modprobe` command is used to load kernel modules manually. With the `-r` option, the command `modprobe -r` can also be used to unload kernel modules. We can remove the vfat module by:

```
# modprobe -r vfat
# lsmod | grep fat
#
```

Note that the *fat* kernel module was loaded while loading the *vfat* kernel module and when the *vfat* module was unloaded by `modprobe`, it's dependency, *fat* module was also unloaded. Thus, `modprobe` also manages loading and unloading the dependencies. To load the *vfat* module again, we use:

```
# modprobe vfat
# lsmod | grep fat
vfat                   17461  0
fat                    65950  1 vfat
```

Normally, this is something we have to rarely do, since the udev process does its work so well once the kernel has detected the hardware. However, this is extremely useful for situations where the kernel module has been edited/updated manually and needs to be reloaded.

16.3 Modifying the Kernel module behavior through modprobe

The information about any specific kernel module can be obtained through the `modinfo` command.

```
# modinfo e1000
filename:
  ↪  /lib/modules/3.10.0-693.11.1.el7.x86_64/kernel/drivers/net/ethernet/intel/e1000/e1000.ko.xz
version:        7.3.21-k8-NAPI
license:        GPL
description:    Intel(R) PRO/1000 Network Driver
author:         Intel Corporation, <linux.nics@intel.com>
rhelversion:    7.4
srcversion:     9E0A112E5D47C996E7C4A58
alias:          pci:v00008086d00002E6Esv*sd*bc*sc*i*
...
alias:          pci:v00008086d00001001sv*sd*bc*sc*i*
alias:          pci:v00008086d00001000sv*sd*bc*sc*i*
depends:
intree:         Y
vermagic:       3.10.0-693.11.1.el7.x86_64 SMP mod_unload modversions
```

```
signer:         CentOS Linux kernel signing key
sig_key:        61:B8:E8:7B:84:11:84:F6:2F:80:D6:07:79:AB:69:2A:49:D8:3B:AF
sig_hashalgo:   sha256
parm:           TxDescriptors:Number of transmit descriptors (array of int)
parm:           RxDescriptors:Number of receive descriptors (array of int)
parm:           Speed:Speed setting (array of int)
parm:           Duplex:Duplex setting (array of int)
parm:           AutoNeg:Advertised auto-negotiation setting (array of int)
...
parm:           copybreak:Maximum size of packet that is copied to a new buffer on
 ↪  receive (uint)
parm:           debug:Debug level (0=none,...,16=all) (int)
```

What's really useful for us is the *parm* information towards the end of the kernel module. While not every kernel module has them, many kernel module do provide the option to set parameters. Thus, the `modinfo` command gives us a list of all the parameters that can be set for a kernel module.

Now, we can see the information about the parameters that are available for the *cdrom* module:

```
# modinfo cdrom
filename:       /lib/modules/3.10.0-693.11.1.el7.x86_64/kernel/drivers/cdrom/cdrom.ko.xz
license:        GPL
rhelversion:    7.4
srcversion:     BE3BD0D17D080229D55B173
depends:
intree:         Y
vermagic:       3.10.0-693.11.1.el7.x86_64 SMP mod_unload modversions
signer:         CentOS Linux kernel signing key
sig_key:        61:B8:E8:7B:84:11:84:F6:2F:80:D6:07:79:AB:69:2A:49:D8:3B:AF
sig_hashalgo:   sha256
parm:           debug:bool
parm:           autoclose:bool
parm:           autoeject:bool
parm:           lockdoor:bool
parm:           check_media_type:bool
parm:           mrw_format_restart:bool
```

An interesting example of a parameter is the *lockdoor* parameter, which when enabled locks the cd tray when the device is mounted. In case of these boolean variables, the value `0 = false; 1 = true`. If however, we try to remove a kernel module that's being used, we get the message:

```
# modprobe -r cdrom
modprobe: FATAL: Module cdrom is in use.
```

If the cdrom weren't in use, the command to set the parameter (stop locking the disk tray when the cdrom is mounted) would be:

```
# modprobe cdrom lockdoor=0
```

16.3.1 Setting kernel module parameters on older Linux versions

There was only one entry point for setting the kernel module parameters on older versions of Linux : `/etc/modprobe.conf`. However, RHEL 7 onwards, this is no longer the case. There are a couple of extra locations for modifying the kernel module parameters.

The folder `/lib/modprobe.d` contains many configuration files. These files contain the default settings for their respective kernel modules. The files are dropped here by the RPMs during installation. Typically, we should avoid any modification in this folder.

To modify the kernel parameters we should choose a different directory: `/etc/modprobe.d`. There is an excellent manpage for the modprobe.d directory, which contains the format for specifying the parameters:

```
options modulename option...
```

Thus, to set the kernel parameter for the lockdoor on cdrom to false, we need to make a file: `/etc/modprobe.d/cdrom.conf` which contains just one line:

```
options cdrom lockdoor=0
```

Since in our case, we can't reload the kernel module since it's in use, the only way to ensure that it works is by rebooting the server with `reboot now`. Finding out if it worked might be problematic on a live system.

For certain modules, we can go to the `/sys/module` directory which contains a subdirectory for every kernel module that's currently loaded. We would want to look for a file called `parameters` that sits in the directory for that module name, and check to see what value is set. However, the cdrom module has no such file.

Next, we can check the wih the `dmesg` command, after filtering it appropriately. Then we filter out the irrelevant stuff with the `grep` command. The grep command can be provided an option `-A` which is followed immediately by the number of lines starting from the matching line should be printed.

```
# dmesg | grep cdrom
[    2.516616] cdrom: Uniform CD-ROM driver Revision: 3.20
# dmesg | grep -A5 cdrom
[    2.516616] cdrom: Uniform CD-ROM driver Revision: 3.20
[    2.516931] sr 2:0:0:0: Attached scsi CD-ROM sr0
[    2.818361] e1000 0000:02:01.0 eth0: (PCI:66MHz:32-bit) 00:0c:29:d6:73:d0
[    2.818369] e1000 0000:02:01.0 eth0: Intel(R) PRO/1000 Network Connection
[    3.137847] random: crng init done
[    3.666093] SGI XFS with ACLs, security attributes, no debug enabled
```

16.4 Tuning kernel behavior through proc

The Linux kernel provides an easy to use interface to optimize kernel parameters, called **proc**. To use it, we have to go to the `/proc` file system. In there we can find current status information about the kernel, and also a `/proc/sys` directory that helps us optimize kernel parameters.

There are several files that convey kernel status information, such as the `/proc/cpuinfo` file that helped us detect if the virtualization flag (*vmx*) was set for the processor. There is also the `/proc/partitions` file that shows us the kernel partition table contents. The `/proc/meminfo` file gives us detailed information about the memory.

In the `/proc/sys` directory, there is a subdirectory for every interface offered by the kernel. Some of the most important ones are : **kernel** - to optimize core kernel functionality, **net** - for networking and **vm** - concerning virtual memory management.

If the module has to be loaded, then the parameters should be edited with /etc/modprobe.d/<moduleName>.conf. However, if they're a part of the main kernel, then there's a chance that there's a `sysctl` setting available for it. In that case, it can be edited via the proc file system. All sysctl settings are available under `/proc/sys` directory.

If we go to the `/proc/sys/kernel` folder, we can see that there are a lot of parameters that the kernel is using. While some of the parameters are quite advanced, some are simple and provide useful information such as the *osrelease and hostname* files.

```
# cat /proc/sys/kernel/osrelease
3.10.0-693.11.1.el7.x86_64
# cat /proc/sys/kernel/hostname
vmPrime.somuVMnet.local
```

We can also change the parameters here. For example, in the `/proc/sys/net/ipv4` directory, there is a file for a parameter called *ip_forward*.

```
# cat ip_forward
1
```

This means that the system is configured to forward packets, i.e., the system is configured as a router. If we want to change this behavior, and want to turn that parameter off, we just use the command `echo 0 > ip\_forward`. Note that changing parameters this way is not persistent.

In the `/proc/sys/vm` directory, there is a file for the swappiness parameter, i.e., the willingness to swap or store a file on the hard disk from the RAM when it's no longer needed. This parameter accepts values between 0 to 100 and if we increase the swappiness, we make the kernel swap out data to disk faster.

An important point of note is that the system parameters in this directory are really advanced, and we should only change the values here if we know exactly what we're doing. This is not a place for *experimentation*, as it may even cause boot to fail.

16.5 Using sysctl

To make the changes that we've made persistent, we have to tune **sysctl**.

16.5.1 sysctl command

The `sysctl` command is used to configure the kernel parameters at runtime. The `sysctl -a` command gives us a list of all the tunable options that are currently set.

```
# sysctl -a
abi.vsyscall32 = 1
crypto.fips_enabled = 0
debug.exception-trace = 1
debug.kprobes-optimization = 1
debug.panic_on_rcu_stall = 0
dev.cdrom.autoclose = 1
dev.cdrom.autoeject = 0
...
vm.vfs_cache_pressure = 100
vm.zone_reclaim_mode = 0
```

This list is also greppable, and thus, we can easily find the parameter we're looking for.

```
# sysctl -a | grep ip_forward
sysctl: reading key "net.ipv6.conf.all.stable_secret"
sysctl: reading key "net.ipv6.conf.default.stable_secret"
sysctl: reading key "net.ipv6.conf.ens33.stable_secret"
sysctl: reading key "net.ipv6.conf.lo.stable_secret"
sysctl: reading key "net.ipv6.conf.virbr0.stable_secret"
sysctl: reading key "net.ipv6.conf.virbr0-nic.stable_secret"
net.ipv4.ip_forward = 0
net.ipv4.ip_forward_use_pmtu = 0
```

The names of the parameters shown here correspond to a file of the same name in the `/proc/sys/` directory. Thus, the `net.ipv4.ip_forward` parameter can be set using the file `/proc/sys/net/ipv4/ip_forward` file.

Upon booting the sysctl process reads a bunch of configuration files to appropriately set the kernel parameters. In earlier versions, there used to be only one config file: `/etc/sysctl.conf`, but on RHEL 7 it's empty. For custom settings, the `/etc/sysctl.d/<name>.conf` files should be used. The default sysctl config is loaded from the `/usr/lib/sysctl.d/` files. The contents of that directory is:

```
# ls /usr/lib/sysctl.d/
00-system.conf  10-default-yama-scope.conf  50-default.conf  60-libvirtd.conf
```

The numbers in front of the file names just ensure that the config settings are read from the files in order. So, *00-system.conf* will be read before *60-libvirtd.conf*. This gives a cascading effect, and if a setting is saved in multiple files then the last file to be read contains the value that'll be used.

Now, if we want our computer to be able to forward IP packets like a router, we edit the file `/etc/sysctl.d/50-ipforward.conf`:

```
net.ipv4.ip_forward = 1
```

After the next reboot, the system will be able to route ip packets. There are certain tools that can write directly to the sysctl config files, but they should be avoided, since to verify that these work, we have to echo their values from the `/proc` file system. Then all we need to do is restart the system, and since the tunable is a part of sysctl system, the parameters will be applied at next boot.

16.6 Updating the kernel

The command to update the kernel is:

```
# yum update kernel
```

If the kernel has already been downloaded, we could use either of the commands below:

```
# yum localinstall <kernelName>.rpm
# rpm -Uvh <kernelName>.rpm
```

In any case, the old kernel is not overwritten during the update, and thus it can still be booted, as both are added to grub.

Chapter 17

Using Kickstart

17.1 Understanding Kickstart Usage

The purpose of using kickstart is to automatically provide the settings we'd normally provide manually to the OS during installation. Now, to use kickstart, we need to provide the kickstart file (which contains the settings) at a place available to the installer.

Let us consider a minimal installation using *boot.iso*. Also, let us assume the kickstart file is named `myks.cfg`. Either USB keys (or any other storage media) or a server could host the file and provide it to the installer. It doesn't matter whether the installation is from a local media (e.g., usb key/ DVD) or a repository server. Once the installation starts, there is nothing more to do for the `SysAdmin` to do but to wait for it to finish, and the kickstart installation needs no manual intervention whatsoever!

17.2 Creating a Kickstart file

If the goal is to install just one server, it is easier to just install it manually, but for a number of servers (with the same configuration), using kickstart is much better. The kickstart file is passed to the installer.

The home directory or user `root` contains two kickstart files:

```
# ls -l /root
total 8
-rw-------. 1 root root 2183 Nov 25 09:09 anaconda-ks.cfg
-rw-r--r--. 1 root root 2214 Nov 25 09:47 initial-setup-ks.cfg
```

To use kickstart with only minor variaitons of the given configuraiton in either of those files, we could make a copy of the file and edit the attributes till they meet our requirements. However, a custom installation by creating a new kickstar file is also possible. For this, we need the `system-config-kickstart` utility (isn't installed by default).

The `system-config-kickstart` command launches the GUI tool that creates the required kickstart files. This GUI has the ability to ask for all the options asked during boot. However, some essential options like the option to create LVM is missing from the kickstart configuration. When done, the options can be saved to the file system.

17.2.1 Installation Scripts

The real power of kickstart comes from the fact that it is capable of running pre-installation and post-installatin scripts.

17.3 Using the Kickstart file for Automatic installations

For network booting using a kickstart file, first we copy the kickstart file to a FTP server.

```
# scp -P 22 automate-install-ks.cfg somu@infraServer.somuVMnet.com:~
# ssh -p 22 root@infraServer.somuVMnet.com
# cp automate-install-ks.cfg /var/ftp/pub
# cd /var/ftp/pub
# chmod 644 automate-install-ks.cfg # Others need read access to be able to use the
↪   kickstart file.
# systemctl status vsftpd # Checking if the FTP server is functioning correctly!
```

Now on the machine where the OS is being installed, when the installer has fully loaded and provides the option to either directly install the OS, or test installation media and then install the OS, we need to press tab to set the installation options.

There, at the end, we need to type provide the location of the kickstart file on the network:

```
ks=ftp://infraServer.somuVMnet.com/ftp/pub/automated-install-ks.cfg
```

Then depending on which options were not given in the kickstart file, the installer may ask for some options, but if all the necessary details are given, then no manual intervention will be required at all!

In cases of data-centers and places where a large number of servers need to be installed, the use of a DVD is impractical and a installation server should be set up.

17.4 Using Kickstart files in fully automated data-centers

There are two servers here - one on the left on which the OS will be installed, and the other on the right, the installation server, which contains the boot image and installer. The server on which the OS will be installed performs a PXE boot (read as *pixie* boot), which is essentially booting from an installation image on the network through the NIC, while a server (installation server) is waiting for it.

On the installation server there is a **DHCP** (Dynamic Host Configuration Protocol) server and a **TFTP** (Trivial File Transfer Protocol). The DHCP server hands out IP addresses and the TFTP server makes the boot image available for download and installation. This boot image is received on the server on which the OS will be installed, and the installer is loaded. On a normal network installation this is where the network's role would end.

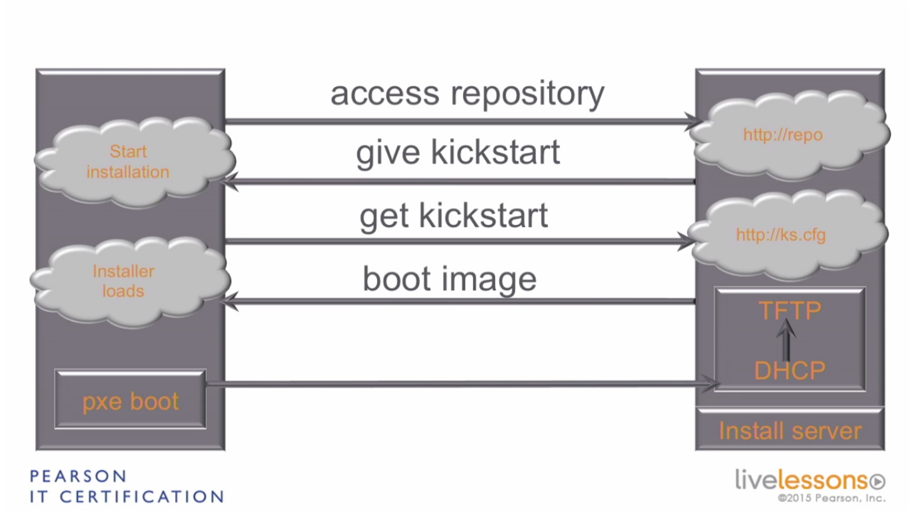

Figure 17.1: PXE (Network) boot with Kickstart

However, since here the OS is installed using a Kickstart file, the client sends a request to receive a copy of the kickstart file from the web/FTP server on the installation server. Upon receiving the kickstart file, finally the installation begins. Finally, the installation needs access to the repository on the installation server for the rpm files that the installer will install (since the installation server is also the installation source). The repository needs to be pre-configured on the installation server as well.

Chapter 18

Managing and Understanding the Boot Procedure

18.1 Boot Procedure Generic Overview

On starting up, the computer perform a "Power On Self Test" (POST). During this, the computer checks all the connected hardware and finds the boot device, which is typically a hard disk / solid state drive (HDD/SSD). On the boot device, the computer access Grub 2, the boot loader, that loads the **kernel** and **initrd**.

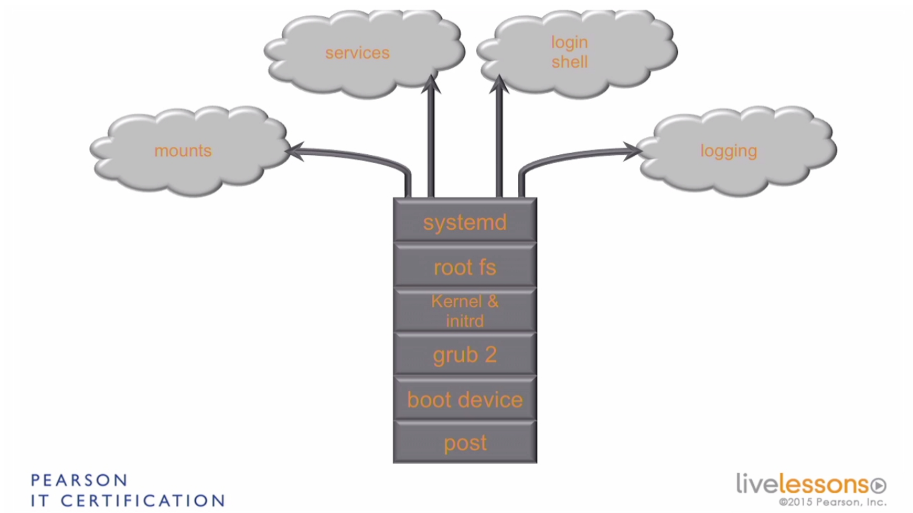

Figure 18.1: Boot Procedure

Next, the root file system is mounted (by the kernel) and then, **systemd** is started. Once systemd has started, everything else can begin, such as: logging, mounting the other file systems, starting all services and preparing the login shell.

18.2 Understanding Grub2

The very first thing from the linux perspective (i.e., the first thing that's executed) when a computer boots is Grub2 (Grand Unified Boot-loader).

The /etc/default/grub is the most important configuration file for Grub2. Most of the customizations/modifications by an user is done to this file. There are also additional configuration files in the /etc/grub.d directory. If any of the configuration files have been updated, the boot loader needs to be updated as well, by using the grub2-mkconfig command. This updates the data in the Master Boot Record (MBR) and the metadata in the first few sectors of our hard drives.

Once the computer boots, we can access the Grub boot menu by pressing the *escape* key. When this is done, we can enter special boot instructions on it.

18.2.1 Booting in emergency mode

On the boot menu, we need to enter systemd.unit=emergency.target as a boot option to start up the computer in emergency mode, which is used in case the computer can't boot normally.

The diagram below explains the entire boot procedure. Once the power is supplied to the computer, it performs the Power On Self Test and then loads the boot loader from the MBR. Now, we have the option to enter the boot menu by pressing the escape key, and enter the boot options, like booting in emergency mode.

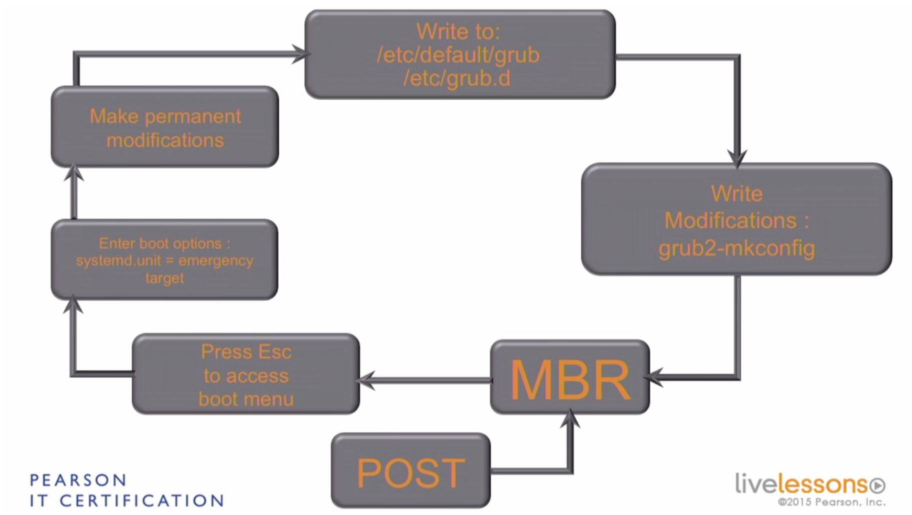

Figure 18.2: Booting in emergency mode

In case there's something wrong with the bootloader itself, we can make permanent modifications by editing the files: /etc/default/grub and the config files in /etc/grub.d directory. Once these modifications have been written, the boot loader needs to be updated using grub2-mkconfig command. This ensures that the next time the MBR will be read, the edited grub2 configuration files will be used.

18.3 Modifying Grub2 Parameters

The primary grub configuration file is /etc/default/grub. The default contents of it look like:

```
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
```

```
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
```

The `GRUB_TIMEOUT` parameter defines the amount of time the system waits in the Grub boot menu for user input. The most important parameter is `GRUB_CMDLINE_LINUX` which defines which arguments are passed on to the linux kernel when the system is booting. The last portion of this parameter, `rhgb quiet` stops Grub from showing us what it's doing during boot. To enable this feature, we need to delete those arguments.

Next, we take a look at the `/etc/grub.d` directory. The files in here are shell scripts that aren't meant to be changed normally, and help with the boot process.

After making all changes, we need to execute the `grub2-mkconfig` command to update the changes to the Grub2 boot-loader. The command reads (and compiles) every grub related config file. This generates a grub configuration file (to send to the boot loader).

To see all these changes, we need to reboot our computer. To verify that the changes have been applied correctly, we can enter the Grub menu using the *Escape* key. In there, we can find the kernel line with all the options that are used. A *CTRL+X* at this point causes a reboot with the new parameters passed to the kernel.

18.4 Understanding Systemd

Systemd is a major new feature added in RHEL 7. It is a new init system that starts things - it both bootstraps the current user-space as well as manage the system processes after booting.

During startup, right after the loading of the kernel, systemd is started, and systemd in turn takes care of starting everything else. Unlike the older *runlevel* system, where only services were started, systemd takes care of services, mounting partitions, auto mounting file systems, and much more.

18.4.1 Unit file

A **unit** file is the replacement of the old init script. Init scripts were relatively more difficult to understand. The unit files have greater readability.

This unit file defines how to start services and everything else systemd can do, as well as define the relation between all those things. Systemd has two different locations for the storing of scripts - the default scripts are stored in `/usr/lib/systemd` and the administrator's custom scripts reside in the `/etc/systemd` directory.

18.5 Managing Services in a systemd Environment

To get a task done in Linux, we need services, which are started using systemd. The directory `/usr/lib/systemd/system` contains many service scripts (among other files). This directory is for the default services that are installed by the RPMs. Thus, we shouldn't edit the files in this directory.

For our own system service management needs, we go to the `/etc/systemd/system` folder. This has two advantages: i) updates to the RPMs that dropped the service scripts in `/usr/lib/systemd/system` won't overwrite our scripts, and ii) Our scripts in `/etc/systemd/system` will overwrite those in the other folder.

18.5.1 Service files

The services form the basic unit of management in systemd is a service. The service files contain all the information required to start a service.

Let us consider the `/usr/lib/systemd/system/httpd.service` file:

```
[Unit]
Description=The Apache HTTP Server
After=network.target remote-fs.target nss-lookup.target
Documentation=man:httpd(8)
Documentation=man:apachectl(8)

[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/httpd
ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
ExecReload=/usr/sbin/httpd $OPTIONS -k graceful
ExecStop=/bin/kill -WINCH ${MAINPID}
# We want systemd to give httpd some time to finish gracefully, but still want
# it to kill httpd after TimeoutStopSec if something went wrong during the
# graceful stop. Normally, Systemd sends SIGTERM signal right after the
# ExecStop, which would kill httpd. We are sending useless SIGCONT here to give
# httpd time to finish.
KillSignal=SIGCONT
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

Due to the usage of systemd, a service in RHEL 7 is much more powerful than a service in previous versions. It is possible to basically turn everything into a service and control it using systemd.

The `[Install]` section of the file defines how the service should be started. The `WantedBy` parameter is defined to set this. Here, the service must be started by a *target*. Services are assigned to targets and the targets take care of starting up the services.

Next, we take a look at the service definition. In earlier versions of RHEL, this section was implemented by the help of large shell scripts. Now, only a few lines of configuration settings are needed. This section defines what should be started and how.

18.5.2 systemctl

Services are managed using the `systemctl` command. For example, to see the status of a service, we write:

```
# systemctl status httpd -l
● httpd.service - The Apache HTTP Server
Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
```

```
Active: active (running) since Sat 2017-12-16 09:31:03 IST; 3s ago
Docs: man:httpd(8)
man:apachectl(8)
Main PID: 5831 (httpd)
Status: "Processing requests..."
CGroup: /system.slice/httpd.service
├─5831 /usr/sbin/httpd -DFOREGROUND
├─5840 /usr/sbin/httpd -DFOREGROUND
├─5842 /usr/sbin/httpd -DFOREGROUND
├─5843 /usr/sbin/httpd -DFOREGROUND
├─5844 /usr/sbin/httpd -DFOREGROUND
└─5845 /usr/sbin/httpd -DFOREGROUND

Dec 16 09:31:01 vmPrime.somuVMnet.local systemd[1]: Starting The Apache HTTP Server...
Dec 16 09:31:03 vmPrime.somuVMnet.local systemd[1]: Started The Apache HTTP Server.
```

To start a service we use:

```
# systemctl start httpd
```

To stop it we use:

```
# systemctl stop httpd
```

To permanently remove the service from the startup procedure of our OS, we use:

```
# systemctl disable httpd
Removed symlink /etc/systemd/system/multi-user.target.wants/httpd.service.
```

To enable the service again:

```
# systemctl enable httpd
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to
↪   /usr/lib/systemd/system/httpd.service.
```

18.5.3 Targets

Our systems can enter different states called **targets**, which are also defined in `/usr/lib/systemd/system` and `/etc/systemd/system`. Targets act as a collection of services, and we can specify dependency relations within the target file. Two of the most important targets are: `multi-user.target` and `graphical.target`, both present in `/usr/lib/systemd/system` directory.

The *graphical.target* is started as the default environment when a GUI is running. Contrastingly, the *multi-user.target* is used as a default environment on servers where a GUI isn't present.

18.5.4 Wants

In order to put services in a specific target, we create a *wants* directory for that target and put a symbolic link to that service in that directory. Services belong to a specific target. When a service is enabled, a symbolic link is created in some *Wants* directory. Each target has its own wants directory that ends with the name of the target followed by a `.wants`. For example, the `multi-user.target` has a corresponding directory called `multi-user.target.wants` in the same folder.

These directories only contain symbolic links to services that should be available at all times in that particular target. For example, the `multi-user.target.wants` contains:

```
# ls -l /usr/lib/systemd/system/multi-user.target.wants/
total 0
lrwxrwxrwx. 1 root root 16 Nov 25 08:50 brandbot.path -> ../brandbot.path
lrwxrwxrwx. 1 root root 15 Nov 25 08:50 dbus.service -> ../dbus.service
lrwxrwxrwx. 1 root root 15 Nov 25 10:14 getty.target -> ../getty.target
lrwxrwxrwx. 1 root root 24 Nov 25 08:50 plymouth-quit.service -> ../plymouth-quit.service
lrwxrwxrwx. 1 root root 29 Nov 25 08:50 plymouth-quit-wait.service -> ../plymouth-quit-wait.service
lrwxrwxrwx. 1 root root 33 Nov 25 10:14 systemd-ask-password-wall.path -> ../systemd-ask-password-wall.path
lrwxrwxrwx. 1 root root 25 Nov 25 10:14 systemd-logind.service -> ../systemd-logind.service
lrwxrwxrwx. 1 root root 39 Nov 25 10:14 systemd-update-utmp-runlevel.service -> ../systemd-update-utmp-runlevel.service
lrwxrwxrwx. 1 root root 32 Nov 25 10:14 systemd-user-sessions.service -> ../systemd-user-sessions.service
```

Further, there are also the services resident in `/etc/systemd/system/multi-user.target.wants` which will also be included:

```
# ls -l /etc/systemd/system/multi-user.target.wants/
total 0
lrwxrwxrwx. 1 root root 41 Nov 25 08:51 abrt-ccpp.service -> /usr/lib/systemd/system/abrt-ccpp.service
lrwxrwxrwx. 1 root root 37 Nov 25 08:50 abrtd.service -> /usr/lib/systemd/system/abrtd.service
lrwxrwxrwx. 1 root root 41 Nov 25 08:50 abrt-oops.service -> /usr/lib/systemd/system/abrt-oops.service
lrwxrwxrwx. 1 root root 43 Nov 25 08:51 abrt-vmcore.service -> /usr/lib/systemd/system/abrt-vmcore.service
lrwxrwxrwx. 1 root root 41 Nov 25 08:50 abrt-xorg.service -> /usr/lib/systemd/system/abrt-xorg.service
lrwxrwxrwx. 1 root root 35 Nov 25 08:59 atd.service -> /usr/lib/systemd/system/atd.service
lrwxrwxrwx. 1 root root 38 Nov 25 08:51 auditd.service -> /usr/lib/systemd/system/auditd.service
lrwxrwxrwx. 1 root root 44 Nov 25 08:59 avahi-daemon.service -> /usr/lib/systemd/system/avahi-daemon.service
lrwxrwxrwx. 1 root root 39 Nov 25 08:51 chronyd.service -> /usr/lib/systemd/system/chronyd.service
lrwxrwxrwx. 1 root root 37 Nov 25 08:50 crond.service -> /usr/lib/systemd/system/crond.service
lrwxrwxrwx. 1 root root 33 Nov 25 08:55 cups.path -> /usr/lib/systemd/system/cups.path
lrwxrwxrwx. 1 root root 36 Nov 25 08:55 cups.service -> /usr/lib/systemd/system/cups.service
lrwxrwxrwx. 1 root root 41 Nov 25 08:51 firewalld.service -> /usr/lib/systemd/system/firewalld.service
lrwxrwxrwx. 1 root root 37 Dec 16 11:32 httpd.service -> /usr/lib/systemd/system/httpd.service
lrwxrwxrwx. 1 root root 42 Nov 25 08:59 irqbalance.service -> /usr/lib/systemd/system/irqbalance.service
lrwxrwxrwx. 1 root root 37 Nov 25 08:51 kdump.service -> /usr/lib/systemd/system/kdump.service
```

```
lrwxrwxrwx. 1 root root 35 Nov 25 08:51 ksm.service ->
  ↪ /usr/lib/systemd/system/ksm.service
lrwxrwxrwx. 1 root root 40 Nov 25 08:51 ksmtuned.service ->
  ↪ /usr/lib/systemd/system/ksmtuned.service
lrwxrwxrwx. 1 root root 46 Nov 25 08:50 libstoragemgmt.service ->
  ↪ /usr/lib/systemd/system/libstoragemgmt.service
lrwxrwxrwx. 1 root root 40 Nov 25 08:52 libvirtd.service ->
  ↪ /usr/lib/systemd/system/libvirtd.service
lrwxrwxrwx. 1 root root 38 Nov 25 08:59 mcelog.service ->
  ↪ /usr/lib/systemd/system/mcelog.service
lrwxrwxrwx. 1 root root 41 Nov 25 08:51 mdmonitor.service ->
  ↪ /usr/lib/systemd/system/mdmonitor.service
lrwxrwxrwx. 1 root root 44 Nov 25 08:59 ModemManager.service ->
  ↪ /usr/lib/systemd/system/ModemManager.service
lrwxrwxrwx. 1 root root 46 Nov 25 08:50 NetworkManager.service ->
  ↪ /usr/lib/systemd/system/NetworkManager.service
lrwxrwxrwx. 1 root root 41 Nov 25 08:52 nfs-client.target ->
  ↪ /usr/lib/systemd/system/nfs-client.target
lrwxrwxrwx. 1 root root 39 Nov 25 08:59 postfix.service ->
  ↪ /usr/lib/systemd/system/postfix.service
lrwxrwxrwx. 1 root root 40 Nov 25 08:50 remote-fs.target ->
  ↪ /usr/lib/systemd/system/remote-fs.target
lrwxrwxrwx. 1 root root 36 Nov 25 08:59 rngd.service ->
  ↪ /usr/lib/systemd/system/rngd.service
lrwxrwxrwx. 1 root root 39 Nov 25 08:59 rsyslog.service ->
  ↪ /usr/lib/systemd/system/rsyslog.service
lrwxrwxrwx. 1 root root 38 Nov 25 08:59 smartd.service ->
  ↪ /usr/lib/systemd/system/smartd.service
lrwxrwxrwx. 1 root root 36 Nov 25 08:59 sshd.service ->
  ↪ /usr/lib/systemd/system/sshd.service
lrwxrwxrwx. 1 root root 39 Nov 25 08:59 sysstat.service ->
  ↪ /usr/lib/systemd/system/sysstat.service
lrwxrwxrwx. 1 root root 37 Nov 25 08:59 tuned.service ->
  ↪ /usr/lib/systemd/system/tuned.service
lrwxrwxrwx. 1 root root 40 Nov 25 08:51 vmtoolsd.service ->
  ↪ /usr/lib/systemd/system/vmtoolsd.service
```

Now, the `/etc/systemd/system/default.target` defines which target (graphical or multi-user) is set as the default environment post-boot for the users.

```
# ls -l /etc/systemd/system/default.target
lrwxrwxrwx. 1 root root 36 Nov 25 09:08 /etc/systemd/system/default.target ->
  ↪ /lib/systemd/system/graphical.target
```

Above, we can see that the graphical.target is set as the default. If we want to change that, we just change the link to point to `/lib/systemd/system/multi-user.target` to operate in a CLI by default.

18.5.5 Viewing Currently Loaded Targets

To view the currently loaded targets we use:

```
# systemctl list-units --type=target
UNIT                   LOAD   ACTIVE SUB    DESCRIPTION
basic.target           loaded active active Basic System
bluetooth.target       loaded active active Bluetooth
cryptsetup.target      loaded active active Encrypted Volumes
```

```
getty.target           loaded active active Login Prompts
graphical.target       loaded active active Graphical Interface
local-fs-pre.target    loaded active active Local File Systems (Pre)
local-fs.target        loaded active active Local File Systems
multi-user.target      loaded active active Multi-User System
network-online.target  loaded active active Network is Online
network-pre.target     loaded active active Network (Pre)
network.target         loaded active active Network
nfs-client.target      loaded active active NFS client services
nss-user-lookup.target loaded active active User and Group Name Lookups
paths.target           loaded active active Paths
remote-fs-pre.target   loaded active active Remote File Systems (Pre)
remote-fs.target       loaded active active Remote File Systems
slices.target          loaded active active Slices
sockets.target         loaded active active Sockets
sound.target           loaded active active Sound Card
swap.target            loaded active active Swap
sysinit.target         loaded active active System Initialization
timers.target          loaded active active Timers

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

22 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

The services provided by our entire OS are not packed together into one monolithic target, but broken down into several targets that concurrently active, as can be seen above. How these targets are supposed to work together is also defined in the target files. For example, in the `/usr/lib/systemd/system/multi-user.target` file:

```
[Unit]
Description=Multi-User System
Documentation=man:systemd.special(7)
Requires=basic.target
Conflicts=rescue.service rescue.target
After=basic.target rescue.service rescue.target
AllowIsolate=yes
```

In this, we can see that the `multi-user.target` requires the *basic.target* to be loaded. It conflicts with *rescue.target* and it has to be loaded only after the *basic.target* is loaded.

Thus, when systemd will try to load the *multi-user.target*, it'll first check the dependencies of the target, which is *basic.target*. If it's not currently loaded, systemd attempts to start the *basic.target* after resolving all of its dependencies, and so on.

18.6 Understanding systemd Targets

Unit files are everything that can be started by systemd. A category of unit files are *targets*. Systemd targets are a collection of unit files that are meant to work together to let the system enter a specific state. Some of these targets are the equivalent of runlevels in the previous versions of RHEL. For example:

Options	Description
poweroff.target	State that shuts down the computer.
rescue.target	Lets the system enter a troubleshooting mode.
multi-user.target	Fully operational server with a command line, but without a GUI.
graphical.target	Fully operational server with a GUI.
reboot.target	State that causes the computer to reboot.
emergency.target	Minimalistic rescue mode, to be used when rescue mode fails.

18.6.1 Services related to targets

The services need to know which target they belong to, and the targets themselves need to know about the ordering. By the use of *wants*, every service knows by which target it is wanted. For example, every service has an Install section containing the name of the target that wants it. The `sshd.service` has:

```
[Install]
WantedBy=multi-user.target
```

Ordering

The order between targets is defined in targets. For example, the *multi-user.target* file contains:

```
[Unit]
Description=Multi-User System
Documentation=man:systemd.special(7)
Requires=basic.target
Conflicts=rescue.service rescue.target
After=basic.target rescue.service rescue.target
AllowIsolate=yes
```

Here, we see that the target (and consequently, the services in it) can only be loaded if the basic.target is already loaded (since it's required). Further, systemd may only attempt to start the services in this target *after* the basic.target has been loaded, and the conflicted *rescue.target* was ordered to load (but didn't). The `AllowIsolate=yes` signifies whether the system can jump from another target to this target to change it's state.

18.7 Switching between systemd Targets

While changing from one system state to another, only certain targets may switch to another one from an operational environment, but in may cases, we can't. For example, it is possible to go from an operational environment to a minimal environment such as the rescue mode.

However, any target can be booted to from the Grub boot menu. The currently active targets can be listed with:

```
# systemctl list-units --type=target
UNIT                   LOAD   ACTIVE SUB    DESCRIPTION
basic.target           loaded active active Basic System
bluetooth.target       loaded active active Bluetooth
cryptsetup.target      loaded active active Encrypted Volumes
```

```
getty.target           loaded active active Login Prompts
graphical.target       loaded active active Graphical Interface
local-fs-pre.target    loaded active active Local File Systems (Pre)
local-fs.target        loaded active active Local File Systems
multi-user.target      loaded active active Multi-User System
network-online.target  loaded active active Network is Online
network-pre.target     loaded active active Network (Pre)
network.target         loaded active active Network
nfs-client.target      loaded active active NFS client services
nss-user-lookup.target loaded active active User and Group Name Lookups
paths.target           loaded active active Paths
remote-fs-pre.target   loaded active active Remote File Systems (Pre)
remote-fs.target       loaded active active Remote File Systems
slices.target          loaded active active Slices
sockets.target         loaded active active Sockets
sound.target           loaded active active Sound Card
swap.target            loaded active active Swap
sysinit.target         loaded active active System Initialization
timers.target          loaded active active Timers

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

22 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

18.7.1 Switching to another target from an operational environment

Working environments consist of multiple targets, some of which are listed above. To change to another target (mode), we use the `systemctl isolate` command:

```
# systemctl isolate rescue.target
Give root password for maintenance
(or type Control-D to continue):
```

To exit rescue mode, we must just type `exit` and let the computer reboot, since it's not possible to easily switch from the rescue mode to any other mode.

18.7.2 Selecting target from Grub Boot menu

When the grub boot menu is displayed, and the available kernels are shown, we can press the `e` key to enter boot options. In here, we have to go down to the line that starts with `linux16` and at the very end, we type: `systemd.unit=<targetName>.target` to boot into it. For example, to boot into the rescue mode during boot, we use:

```
systemd.unit=rescue.target
```

Then, we have to press *CTRL+X* to execute. This will directly boot us into the rescue mode. In this mode, the `systemctl list-units --type=target` returns only a few targets, which proves that this mode is indeed minimalistic, but also the loaded targets (i.e., the services loaded by them) are essential for proper functioning of the computer.

18.7.3 Emergency mode

To boot into the emergency mode we need to use `systemctl.unit=emergency.target`. In this mode, `systemctl list-units --type=targets` doesn't return anything. We can use `systemctl default` to start the default target.

18.8 Managing File System mounts in a systemd Environment

Other than using `/etc/fstab`, systemd also provides a way to mount file systems. Further, not all file systems are mounted (or available) using `/etc/fstab`. The file systems that can be mounted using systemd (called **mount units**) can be obtained by:

```
# ls *.mount
dev-hugepages.mount           sys-kernel-config.mount
dev-mqueue.mount              sys-kernel-debug.mount
proc-fs-nfsd.mount            tmp.mount
proc-sys-fs-binfmt_misc.mount var-lib-nfs-rpc_pipefs.mount
sys-fs-fuse-connections.mount
```

These contain the specifications for certain file systems that need to be mounted at all times, such as `/tmp`. The contents of `tmp.mount` is:

```
[Unit]
Description=Temporary Directory
Documentation=man:hier(7)
Documentation=http://www.freedesktop.org/wiki/Software/systemd/APIFileSystems
ConditionPathIsSymbolicLink=!/tmp
DefaultDependencies=no
Conflicts=umount.target
Before=local-fs.target umount.target
After=swap.target

[Mount]
What=tmpfs
Where=/tmp
Type=tmpfs
Options=mode=1777,strictatime

# Make 'systemctl enable tmp.mount' work:
[Install]
WantedBy=local-fs.target
```

While the unit specification is very generic, the `[Mount]` and `[Install]` specifications are very important. The `What` defines the file system to be mounted, the `Where` clause defines the location to mount the file system. The file system type is *tmpfs* and there are certain mount options as well. The *Install* section defines that `local-fs.target` needs this mount point to work, which in turn makes it possible to mount this file system using `systemctl`.

If we want a custom mount file like this, we have to put it in `/etc/systemd/system` directory. A bare-bones mount unit file would look like:

```
# Mount unit for /dev/vgPrime/lvPrime

```

```
[Unit]
Description="My test mount"

[Mount]
what=/dev/vgPrime/lvPrime
Where=/myLv
Type=xfs

[Install]
WantedBy=multi-user.target
```

Then we mount the disk and check its status using:

```
# systemctl start myLv.mount
# systemctl status myLv.mount -l
● myLv.mount - "My test mount"
Loaded: loaded (/etc/systemd/system/myLv.mount; disabled; vendor preset: disabled)
Active: active (mounted) since Wed 2017-12-20 10:51:59 IST; 3s ago
Where: /myLv
What: /dev/mapper/vgPrime-lvPrime
Process: 3510 ExecMount=/bin/mount /dev/vgPrime/lvPrime /myLv -t xfs (code=exited,
↪ status=0/SUCCESS)

Dec 20 10:51:59 vmPrime.somuVMnet.com systemd[1]: Mounting "My test mount"...
Dec 20 10:51:59 vmPrime.somuVMnet.com systemd[1]: Mounted "My test mount".
```

Now, to ensure that the disk is auto-mounted when the multi-user.target is loaded, we need to add a symlink to it in the *wants* directory for that target. This is achieved by:

```
# systemctl enable myLv.mount
Created symlink from /etc/systemd/system/multi-user.target.wants/myLv.mount to
↪ /etc/systemd/system/myLv.mount.
```

This makes sure that every time the *multi-user.target* is active, the file system *myLv* is auto-mounted.

18.9 Managing Automount in a systemd Environment

To auto-mount a file system, the procedure is similar to manually mounting a file system. Just like the latter, we need to create a *Mount unit file* for the file system. Then, we want the file system to be mounted whenever a certain activity occurs in the auto-mount directory. To do this with *myLv.mount*, we first need to disable it. Once that is done, we also need to disconnect the mount.

```
# systemctl disable myLv.mount
Removed symlink /etc/systemd/system/multi-user.target.wants/myLv.mount.
# systemctl stop myLv.mount
# systemctl status myLv.mount
● myLv.mount - "My test mount"
Loaded: loaded (/etc/systemd/system/myLv.mount; disabled; vendor preset: disabled)
Active: inactive (dead)
Where: /myLv
What: /dev/vgPrime/lvPrime

Dec 20 10:51:59 vmPrime.somuVMnet.com systemd[1]: Mounting "My test mount"...
```

```
Dec 20 10:51:59 vmPrime.somuVMnet.com systemd[1]: Mounted "My test mount".
Dec 20 11:08:37 vmPrime.somuVMnet.com systemd[1]: Unmounting "My test mount"...
Dec 20 11:08:38 vmPrime.somuVMnet.com systemd[1]: Unmounted "My test mount".
```

18.9.1 Automount Unit file

The auto-mounting of a directory needs an auto-mount unit file, which is named following the syntax: `<mountFileName>.automount`. Since our LV has a mount file called *myLv.mount*, we have to use the name `myLv.automount`. The automount unit file is relatively much simpler than its manual mounting counterpart.

```
[Unit]
Description = myLv Automount

[Automount]
Where = /myLv

[Install]
WantedBy = multi-user.target
```

At this point, we can enable and start the automount unit:

```
# systemctl enable myLv.automount
Created symlink from /etc/systemd/system/multi-user.target.wants/myLv.automount to
↪ /etc/systemd/system/myLv.automount.
# systemctl start myLv.automount
# systemctl status myLv.automount
● myLv.automount - myLv Automount
Loaded: loaded (/etc/systemd/system/myLv.automount; enabled; vendor preset: disabled)
Active: active (waiting) since Wed 2017-12-20 11:21:51 IST; 7s ago
Where: /myLv

Dec 20 11:21:51 vmPrime.somuVMnet.com systemd[1]: Set up automount myLv Automount.
Dec 20 11:21:51 vmPrime.somuVMnet.com systemd[1]: Starting myLv Automount.
```

Another (quicker) way to verify the automount would be to use:

```
# mount | grep myLv
systemd-1 on /myLv type autofs
↪ (rw,relatime,fd=33,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=13344)
/dev/mapper/vgPrime-lvPrime on /myLv type xfs
↪ (rw,relatime,seclabel,attr2,inode64,noquota)
```

18.9.2 Difference between enabling Mount vs Automount Units

The act of enabling the *.mount* files ensures that the `autofs` process mounts the file, while when we enable the *.automount* files makes systemd mount the files. Thus, the latter is the method that is preferred, since systemd is a newer system of initializing services and automounts, and is future-proof.

Chapter 19

Applying Essential Troubleshooting Skills

19.1 Making Grub Changes persistent

19.1.1 Changes made during boot

After making changes in the boot menu, when we finally boot, we can make those changes persistent by rewriting the `/boot/grub2/grub.cfg` file.

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-693.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-693.el7.x86_64.img
Found linux image: /boot/vmlinuz-0-rescue-5cbfb880c0aa466ca7e3be91308fde5f
Found initrd image: /boot/initramfs-0-rescue-5cbfb880c0aa466ca7e3be91308fde5f.img
done
```

19.1.2 Changes made in Configuraiton File

The `/etc/default/grub` file is the configuration file for Grub2 that provides several boot options. These can be changed to affect several boot parameters, and the changes saved to the bootloader. There are also shell scripts in the `/etc/grub.d` directory that aren't meant to be touched by an administrator. These control grub boot procedure as well. Almost all the functionality that we need from grub is provided by a set of grub2 commands:

```
# grub2-
grub2-bios-setup           grub2-mkpasswd-pbkdf2
grub2-editenv              grub2-mkrelpath
grub2-file                 grub2-mkrescue
grub2-fstest               grub2-mkstandalone
grub2-get-kernel-settings  grub2-ofpathname
grub2-glue-efi             grub2-probe
grub2-install              grub2-reboot
grub2-kbdcomp              grub2-rpm-sort
grub2-menulst2cfg          grub2-script-check
grub2-mkconfig             grub2-set-default
```

```
grub2-mkfont               grub2-setpassword
grub2-mkimage              grub2-sparc64-setup
grub2-mklayout             grub2-syslinux2cfg
grub2-mknetdir
```

These commands can be used to accomplish tasks with grub such as install grub (grub2-install), make a new boot image (grub2-mkimage), set a grub boot password (grub2-mkpasswd-pbkdf2), to probe operating system configuration (grub2-probe), to reboot a specific boot image (grub2-reboot) and much more.

19.2 Using rd.break to Reset the Root Password

While on the previous versions of RHEL, resetting the root password or logging on to a system where the root password isn't know was relatively easy. After the introduction of systemd, breaking into the system is a lot harder to do.

First we have to enter the line rd.break and pass it as a kernel parameter in the boot menu (at the end of the kernel line). The **rd.break** parameter instructs the next part of the boot procedure, **initrd**, to break at a specific location of the *initramfs*. This brings us to a system where all the supporting modules are available, but no file system has yet been mounted. This parameter bring us to a root shell without prompting for a root password.

We're in such an early point in the boot procedure that the system root hasn't been mounted to the usual / location yet, and is available at /sysroot in read-only mode. Now, we need to mount the system root in a read-write mode using:

```
# mount -o remount,rw /sysroot
```

Next, we make the content of /sysroot the current root directory using:

```
# chroot /sysroot
```

Now, we simply echo the new password to the passwd utility and reset the password for the user root. The syntax is: echo <newPassword> | passwd --stdin root. The root password thus has to be reset using the command:

```
# echo secret | passwd --stdin root
Changing password for the user root.
passwd: all authentication tokens updated successfully.
# touch /.autorelabel
```

Finally, in the last line, we instruct SELinux to auto-relable. Since we're so early in the boot procedure, SELinux isn't functional, and if we skip this command, our changes will be lost. Now, at this point, it is safe to *CTRL+D* a couple of times and let the OS reboot itself. Once done, we can enter the OS using the root password we just set (*secret* in our case). Now, after the reboot, we can login to the system as root using the new root password.

Part IV

Managing Network Services

Chapter 20

Managing HTTP Services

20.1 Understanding Apache Configuration

The **HTTP Daemon (httpd)** is the apache web server process. To find out more about the process, we use:

```
# which httpd
/usr/sbin/httpd
# rpm -qf /usr/sbin/httpd # Obtaining the name of the package which installed httpd.
httpd-2.4.6-67.el7.centos.6.x86_64
# rpm -qc httpd
/etc/httpd/conf.d/autoindex.conf
/etc/httpd/conf.d/userdir.conf
/etc/httpd/conf.d/welcome.conf
/etc/httpd/conf.modules.d/00-base.conf
/etc/httpd/conf.modules.d/00-dav.conf
/etc/httpd/conf.modules.d/00-lua.conf
/etc/httpd/conf.modules.d/00-mpm.conf
/etc/httpd/conf.modules.d/00-proxy.conf
/etc/httpd/conf.modules.d/00-systemd.conf
/etc/httpd/conf.modules.d/01-cgi.conf
/etc/httpd/conf/httpd.conf
/etc/httpd/conf/magic
/etc/logrotate.d/httpd
/etc/sysconfig/htcacheclean
/etc/sysconfig/httpd
```

The last command, `rpm -qc httpd` shows us the configuration files for the *httpd* process. There are some config files for httpd in `/etc/sysconfig` directory and some in `/etc/httpd` directory.

The `/etc/sysconfig` directory has a file called *httpd* which has some basic configuration for the web server, and this can be used to manage start-up parameters for apache. Thus, whenever there needs to be anything different whie starting the apache web server, this file should be edited.

The important part of the `httpd` configuration is stored in `/etc/httpd`. It's contents are:

```
# ls -l /etc/httpd
total 0
drwxr-xr-x. 2 root root  37 Dec 20 15:36 conf
```

```
drwxr-xr-x. 2 root root  82 Dec 20 15:36 conf.d
drwxr-xr-x. 2 root root 146 Dec 20 15:36 conf.modules.d
lrwxrwxrwx. 1 root root  19 Dec 20 15:36 logs -> ../../var/log/httpd
lrwxrwxrwx. 1 root root  29 Dec 20 15:36 modules -> ../../usr/lib64/httpd/modules
lrwxrwxrwx. 1 root root  10 Dec 20 15:36 run -> /run/httpd
```

The most important of the configuration files is stored in `/etc/httpd/conf/httpd.conf`. It contains all the parameters that might need to be changed to customize the configuration of our apache environment. Some of the important parameter passed to the web server from this file are:

Options	Description
Listen 80	Tells the HTTP server which port to *listen on* (i.e., wait for incoming TPC connections) for HTTP Services.
include conf.modules.d/*.conf	Loads the contents of the `conf.modules.d` directory.

The inclusion of the `/etc/httpd/conf.modules.d` directory is due to the fact that apache has a modular configuration. Both files in the *conf.modules.d* and *conf.d* are included in this configuration. The contents of the `conf.d` are:

```
# ls /etc/httpd/conf.d
autoindex.conf  README  userdir.conf  welcome.conf
```

Some RPMs that deal with the apache web server sometimes drop configuration files in this directory that add another branch of functionality to our web server. This particular folder, *conf.d* is used to house generic configurations. The folder *conf.modules.d* however, contains the configuration for several modules. These include things like the proxy module.

Sometimes, the apache update may cause a new version of the *httpd.conf* to appear, in which case the user config will still be available at *httpd.conf.rpmsave* in the same directory. This is not something specific to apache - yum does this to any config file that has to be overwritten in the process of an upgrade.

20.2 Creating a Basic Web Site

One of the most important configuration settings in the `httpd.conf` file is the **DocumentRoot**, which sets the directory under which all the requests for documents are served. If the DocumentRoot is changed, certain settings in SELinux need to be changed as well! The default value of this is set to `/var/www/html`.

Let us put a basic html file inside this directory:

```html
<html>
<head>
<title>Homepage!</title>
<head>
<body>
<h1>Hello, World!</h1>
<body>
</html>
```

To view this, (even during an SSH session) we can use *elinks*, which is a text based browser. First, we have to start the HTTP daemon (and enable it so that it auto-starts after each reboot):

```
# systemctl start httpd
# systemctl enable httpd
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to
↪ /usr/lib/systemd/system/httpd.service.
# systemctl status -l httpd
● httpd.service - The Apache HTTP Server
Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
Active: active (running) since Wed 2017-12-20 17:26:08 IST; 11s ago
Docs: man:httpd(8)
man:apachectl(8)
Main PID: 5802 (httpd)
Status: "Total requests: 0; Current requests/sec: 0; Current traffic:   0 B/sec"
CGroup: /system.slice/httpd.service
├─5802 /usr/sbin/httpd -DFOREGROUND
├─5806 /usr/sbin/httpd -DFOREGROUND
├─5807 /usr/sbin/httpd -DFOREGROUND
├─5808 /usr/sbin/httpd -DFOREGROUND
├─5809 /usr/sbin/httpd -DFOREGROUND
└─5810 /usr/sbin/httpd -DFOREGROUND

Dec 20 17:26:08 vmPrime.somuVMnet.com systemd[1]: Starting The Apache HTTP Server...
Dec 20 17:26:08 vmPrime.somuVMnet.com systemd[1]: Started The Apache HTTP Server.
```

The contents of the `/var/www/html` directory are now available on localhost. To view the webpages, we use:

```
# elinks http://localhost
```

Chapter 21

Managing SELinux

21.1 Understanding the Need for SELinux

SELinux stands for **Security Enhanced Linux**. Let us consider an application that runs on a server, that provides a backdoor to an attacker who can start a shell session on the server. This can be done as the httpd user in the case of a vulnerability on the web server. Let us consider the attacker uses the /tmp directory (which has rwxrwxrwx permissions) for nefarious purposes. Now, we can't take away permissions, since some applications depend on the directory to have universal permissions. We also can't use a firewall, since it'd block access to HTTP services. Finally, we can't mount the file system with a `NOEXEC` flag (which prevents the execution of scripts on that disk) since sometimes applications use the `/tmp` directory to execute scripts.

Under such circumstances, SELinux becomes extremely necessary, since it permits us to set policies that define exactly what kind of access each application has, and on which directories. Thus, it is critical to use SELinux on any server that is connected to the internet.

21.1.1 SELinux and Syscalls

Every operation on the server is occurring via syscalls. When enabled, all of the syscalls are filtered through SELinux. SELinux can be in either *enforcing* or *permissive* mode for this. Each system calls go through an analysis against a policy that check whether the actions are permitted. Let's assume the action is not permitted, and a *avc:denied* is returned. Now, several things will happen.

First, the event will go through **auditd**, and in any case, whenever SELinux is enabled, *auditd* (configurable via `/etc/audit/auditd.conf`) will write the event to the audit log (`/var/log/audit/audit.log`). This is a very important source of information.

From there, if SELinux is set to *enforcing mode*, the syscall will be immediately stopped. However, in *permissive mode*, it'll go on, since in permissive mode, everything is logged by auditd, but nothing is stopped. Thus, the permissive mode allows us to analyse what is going on without stopping syscalls, stopping which might lead to system crashes and other unforeseen events.

Let us consider another example, where we have a webserver running on localhost, which we try to access using `elinks`. Now, let the webserver's DocumentRoot be set to */blah* directory. `ls -Z` prints the security context of every file or directory. On executing this command on */blah*, we will probably find that the directory has the wrong label.

Now when elinks tries to access the index file on the */blah* directory, it'll generate a *getattr* system call. If SELinux is in enforcing mode, that'll be stopped immediately.

21.2 Understanding SELinux Modes and Policy

To configure SELinux at a basic level, there are three things that we need to understand. The first of them is the SELinux Mode.

21.2.1 SELinux Mode

The SELinux mode is obtained from a file called `/etc/sysconfig/selinux`. There are three possible modes for this: Enforcing, Permissive and disabled. The disabled mode can only be specified while booting. This completely disables SELinux by ensuring all the SELinux libraries that are normally loaded by the kernel won't be loaded at all. In fact, the difference is so drastic, it's not possible to switch between disabled and any other mode without rebooting.

However, it is perfectly fine to toggle between enforcing and permissive modes. The current SELinux mode is given by:

```
# getenforce
Enforcing
```

To change the SELinux mode, we use the command:

```
# setenforce Permissive
# getenforce
Permissive
```

Toggling between the permissive and enforcing modes can be extremely useful for troubleshooting. Let us consider a scenario where we're setting up an FTP server, and it doesn't work. This may be due to an error in the FTP server config, or it's being blocked by SELinux. To make sure SELinux is not at fault, we switch to Permissive mode using `setenforce Permissive` and try again. If it starts working, it was being blocked by SELinux. Then we know where to look for the solution. However, under all other circumstances, the SELinux mode should be set to enforcing.

21.2.2 Context and Policies

Everything on RHEL 7 has a context, which can be displayed by the command:

```
# ls -Z
drwxr-xr-x. somu somu unconfined_u:object_r:user_home_t:s0 Desktop
drwxr-xr-x. somu somu unconfined_u:object_r:user_home_t:s0 Documents
drwxr-xr-x. somu somu unconfined_u:object_r:user_home_t:s0 Downloads
drwxr-xr-x. somu somu unconfined_u:object_r:audio_home_t:s0 Music
drwxr-xr-x. somu somu unconfined_u:object_r:user_home_t:s0 Pictures
drwxr-xr-x. somu somu unconfined_u:object_r:user_home_t:s0 Public
drwxr-xr-x. somu somu unconfined_u:object_r:user_home_t:s0 Templates
drwxr-xr-x. somu somu unconfined_u:object_r:user_home_t:s0 Videos
```

There are three parts to a context, with the delimiter `:` separating them. The first is the *user* part, which is only for advanced SELinux configurations. Next comes the *role* part, which again, is for advanced SELinux configurations. Finally, we have the *type* part. This denotes the kind of access that is allowed to files/directories.

Not only are there contexts on files, there are contexts on processes as well, which can be viewed using `ps Zau`. Even ports have context labels, viewed by using `netstat Ztulpen`. So, the idea is that every file/process/port's context is matched to a policy to grant/deny access.

21.2.3 Booleans

Booleans are easy switches to enable or disable functionalities in a policy. A list of all available booleans can be obtained with:

```
# getsebool -a
abrt_anon_write --> off
abrt_handle_event --> off
abrt_upload_watch_anon_write --> on
antivirus_can_scan_system --> off
...
zebra_write_config --> off
zoneminder_anon_write --> off
zoneminder_run_sudo --> off
```

We can filter the list and find only booleans that have 'ftp' in their boolean name using:

```
# getsebool -a | grep ftp
ftpd_anon_write --> off
ftpd_connect_all_unreserved --> off
ftpd_connect_db --> off
ftpd_full_access --> off
ftpd_use_cifs --> off
ftpd_use_fusefs --> off
ftpd_use_nfs --> off
ftpd_use_passive_mode --> off
httpd_can_connect_ftp --> off
httpd_enable_ftp_server --> off
tftp_anon_write --> off
tftp_home_dir --> off
```

For example, let us consider the boolean `ftpd_full_access --> off` which doesn't allows ftp servers to login to local user accounts and have read/write access to all files subject to Discretionary Access Control (DAC) mechanisms (permissions, ACLs, etc.). Another such boolean is `ftp_home_dir --> off` which doesn't allow users to login to their home directories.

When certain functionalities are turned off, we should always check if some boolean is turned off. In this case, since `ftp_home_dir` is off, the users won't be able to login to their home directories even though *vsftpd* may be configured to do so, since SELinux will prevent it.

21.3 Understanding SELinux Lables and Booleans

To manage SELinux, we need to be able to manage context. The context of the *httpd* process can be viewed with:

```
# ps Zaux | grep httpd
system_u:system_r:httpd_t:s0    root       1249  0.1  0.1 226240  5156 ?        Ss   10:32   0:00 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0    apache     1435  0.0  0.0 228324  3160 ?        S    10:32   0:00 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0    apache     1436  0.0  0.0 228324  3160 ?        S    10:32   0:00 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0    apache     1438  0.0  0.0 228324  3160 ?        S    10:32   0:00 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0    apache     1441  0.0  0.0 228324  3160 ?        S    10:32   0:00 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0    apache     1443  0.0  0.0 228324  3160 ?        S    10:32   0:00 /usr/sbin/httpd -DFOREGROUND
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 root 2738 0.0  0.0 112664 968 pts/0    S+ 10:34   0:00 grep --color=auto httpd
```

Here we can see that the current context of the httpd process is *httpd_t*. Now, the default document root for the httpd process is `/var/www` and we can see its context using:

```
# ls -Z /var/www
drwxr-xr-x. root root system_u:object_r:httpd_sys_script_exec_t:s0 cgi-bin
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 html
```

We can see that the context for `/var/www` has been set correctly. The policy will state that the source context *httpd_t* is allowed to get through the target context *httpd_sys_content_t*.

Now if an attacker finds a vulnerability on a web server script, and tries to access the `/tmp` directory, SELinux would prevent that because the context of `/tmp` has been set to:

```
# ls -dZ /tmp
drwxrwxrwt. root root system_u:object_r:tmp_t:s0       /tmp
```

Thus, the process with the source context *httpd_t* won't be allowed to access a directory with a target context of *tmp_t*.

There are primarily two situations where administrators may need to manage context:

- A file has been moved instead of copied, or
- We want to do something that doesn't correspond to the defaults.

21.3.1 File being moved instead of copied

Let us consider a file *myFile* in our home directory. In that case, it'd have the context of:

```
# touch myFile
# ls -Z
-rw-------. root root system_u:object_r:admin_home_t:s0 anaconda-ks.cfg
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 initial-setup-ks.cfg
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 myFile
```

```
# ls -dZ
dr-xr-x---. root root system_u:object_r:admin_home_t:s0 .
```

We can see that the file we created has a context of `admin_home_t`. This is because the current directory `/root` also has a context of `admin_home_t`.

Now, let us make a copy of the `/etc/hosts` file and name it `/etc/hosts2`. If we move that file, instead of copy it to the home directory of the home user, it'll have a context of:

```
# ls -Z hosts2
-rw-r--r--. root root unconfined_u:object_r:etc_t:s0   hosts2
```

The context of *hosts2* is set to *etc_t* because while moving a file, the original context moves with it. When copying a file, however, a new file is created and it normally inherits the context of the parent (target) directory.

21.3.2 semanage

The **semanage** utility is used to set context. It works with a set of arguments, and a specific argument defines what it's actions will be. Some of the important arguments are:

Options	Description
fcontext	Manages the fcontext of the object.
boolean	Used to change the value of a boolean
port	Changes the port type definition.

The documentation for `semanage` has been arranged in such a way that a separate man page exists for each of the arguments. Thus, there's a man page for `man semanage-fcontext`, `man semanage-boolean`, etc. The examples in the man page for *semanage-fcontext* has examples for setting the context for everything under the `/web` directory:

```
# semanage fcontext -a -t httpd_sys_content_t "/web(/.*)?"
```

The `-t` flag sets the type of `httpd_sys_content_t` for all items that match the regular expression `/web(/.*)?`. This matches everything in the web directory, and any files/sub-directories contained within it.

Note that `semanage fcontext` doesn't write to the file system directly, but to the policy. This is because all the default policies should be set in the policy and not the file system. To apply these changes from the policy to the file system, we need to use the command:

```
# restorecon -R -v /web
```

The `-R` flag makes it recursive and the `-v` flag makes it verbose. The `restorecon` utility is also very useful when something goes wrong with a context, because it checks the policy and ensures that the context of every file in a directory matches their context as described in the policy.

The file we moved from the `/etc/hosts` directory has the wrong context of *etc_t*, instead of *admin_home_t*. This can be fixed using:

```
# restorecon -v hosts2
restorecon reset /root/hosts2 context
↪  unconfined_u:object_r:etc_t:s0->unconfined_u:object_r:admin_home_t:s0
```

```
# ls -Z hosts2
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 hosts2
```

We can see that the file *hosts2* now has the correct context for the directory `/root`. This could also have been done directly on the `/root` directory to fix all the wrong contexts in the directory at once, using `restorecon -R -v /root`.

21.4 Understanding File System Labels

If we want to change the context using `semanage fcontext`, we should know which context to use. There are many contexts to choose from. One possible solution is to go to the target directory and view which context the files use. For example, the contents of `/var/www` directory use:

```
# ls -Z
drwxr-xr-x. root root system_u:object_r:httpd_sys_script_exec_t:s0 cgi-bin
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 html
```

The available contexts are: *httpd_sys_script_exec_t* and *httpd_sys_content_t*. A list of all possible contexts can be displayed using `semanage fcontext -l`. However, it's a long list and grepping does help, but the filtered contents are still long:

```
# semanage fcontext -l | grep http
/usr/.*\.cgi                                      regular file
 ↪  system_u:object_r:httpd_sys_script_exec_t:s0
/opt/.*\.cgi                                      regular file
 ↪  system_u:object_r:httpd_sys_script_exec_t:s0
/srv/([^/]*/)?www(/.*)?                           all files
 ↪  system_u:object_r:httpd_sys_content_t:s0
/srv/([^/]*/)?www/logs(/.*)?                      all files
 ↪  system_u:object_r:httpd_log_t:s0
/var/www(/.*)?                                    all files
 ↪  system_u:object_r:httpd_sys_content_t:s0
/var/www(/.*)?/logs(/.*)?                         all files
 ↪  system_u:object_r:httpd_log_t:s0
...
/usr/share/wordpress-mu/wp-config\.php            regular file
 ↪  system_u:object_r:httpd_sys_script_exec_t:s0
/usr/share/munin/plugins/http_loadtime            regular file
 ↪  system_u:object_r:services_munin_plugin_exec_t:s0
/usr/share/system-config-httpd/system-config-httpd regular file
 ↪  system_u:object_r:bin_t:s0
```

What do help are the SELinux man pages. On previous versions of RHEL, the man pages were available through the command `man -k _selinux`. However, on RHEL 7 these need to be generated by us using the `sepolicy` utility, which isn't installed by default. We can find which package provides it using:

```
# yum provides */sepolicy
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
* base: centos.mirror.net.in
* extras: centos.mirror.net.in
* updates: centos.mirror.net.in
policycoreutils-devel-2.5-17.1.el7.i686 : SELinux policy core policy devel utilities
```

```
Repo        : base
Matched from:
Filename    : /usr/share/bash-completion/completions/sepolicy
Filename    : /usr/bin/sepolicy

policycoreutils-devel-2.5-17.1.el7.x86_64 : SELinux policy core policy devel utilities
Repo        : base
Matched from:
Filename    : /usr/share/bash-completion/completions/sepolicy
Filename    : /usr/bin/sepolicy

policycoreutils-python-2.5-17.1.el7.x86_64 : SELinux policy core python utilities
Repo        : base
Matched from:
Filename    : /usr/lib64/python2.7/site-packages/sepolicy
```

So, we need the policycoreutils development version. So, we install it using the command `yum install -y policycoreutils-devel`.

Once installed, we need to run a command that helps us find the correct man page for a SELinux command, which is:

```
# sepolicy manpage -a -p /usr/share/man/man8
/usr/share/man/man8/NetworkManager_selinux.8
/usr/share/man/man8/abrt_selinux.8
/usr/share/man/man8/abrt_dump_oops_selinux.8
...
/usr/share/man/man8/zoneminder_script_selinux.8
/usr/share/man/man8/zos_remote_selinux.8
# mandb
```

The command generates a list of manpages. Every service available on SELinux has its own manpage, created by running this command. Once the manpages have been generated, we should run the `mandb`, which updates the index of the manpages, making them searchable using `man -k`.

So, to search for the SELinux manpages for anything concerning httpd, we use:

```
# man -k _selinux | grep http
apache_selinux (8)   - Security Enhanced Linux Policy for the httpd processes
httpd_helper_selinux (8) - Security Enhanced Linux Policy for the httpd_helper processes
httpd_passwd_selinux (8) - Security Enhanced Linux Policy for the httpd_passwd processes
httpd_php_selinux (8) - Security Enhanced Linux Policy for the httpd_php processes
httpd_rotatelogs_selinux (8) - Security Enhanced Linux Policy for the httpd_rotatelogs processes
httpd_selinux (8)    - Security Enhanced Linux Policy for the httpd processes
httpd_suexec_selinux (8) - Security Enhanced Linux Policy for the httpd_suexec processes
httpd_sys_script_selinux (8) - Security Enhanced Linux Policy for the httpd_sys_script processes
httpd_unconfined_script_selinux (8) - Security Enhanced Linux Policy for the httpd_unconfined_script processes
httpd_user_script_selinux (8) - Security Enhanced Linux Policy for the httpd_user_script processes
```

Inside these manpages, all booleans and contexts are defined. So, we have a place to look up the appropriate context for the kind of access that our files/processes need.

21.5 Understanding semanage fcontext and chcon differences

In certain man page entries, we might come across the command `chcon`, which is a *bad* program, and shouldn't be used. For this, we need to understand the difference between `semanage`, `fcontext` and `chcon`.

Let us consider a scenario where we need to change the context of a file `/blah/index.html`. Suppose we want to set its context to *httpd_sys_content_t*. To do this using `chcon`, we would need to use the command:

```
# chcon -R --type=httpd_sys_content_t /blah
```

What this command does is set the given context type to the inode, i.e., applies the change to the file system. The corresponding entry for it in the policy still remains *default_t*. This is bad because whenever a relabel operation occurs (typically on the entire root file system [relabel of /]), the context for the `/blah` directory would be overwritten to *default_t*, because during a relabel everything in the policy overwrites everything in the file system. Thus, it is absolutely critical that SELinux information is always written to the policy first! This is why to set the context of a file/directory, we use:

```
# semanage fcontext -a -t httpd_sys_content_t "/blah(/.*)?"
```

This sets the context in the policy and thus the change will survive the relabel activity.

21.6 Using Booleans

To handle booleans, we need two commands: `getsebool` and `setsebool`. The command to list all possible SELinux Boolean Switches on a particular system is given by:

```
# getsebool -a
abrt_anon_write --> off
abrt_handle_event --> off
abrt_upload_watch_anon_write --> on
...
zoneminder_anon_write --> off
zoneminder_run_sudo --> off
```

To find an appropriate boolean for something (e.g., FTP), we use grepping:

```
# getsebool -a | grep ftp
ftpd_anon_write --> off
ftpd_connect_all_unreserved --> off
ftpd_connect_db --> off
ftpd_full_access --> off
ftpd_use_cifs --> off
ftpd_use_fusefs --> off
ftpd_use_nfs --> off
ftpd_use_passive_mode --> off
httpd_can_connect_ftp --> off
httpd_enable_ftp_server --> off
tftp_anon_write --> off
tftp_home_dir --> off
```

For example, if we want to turn on the switch for `ftpd_use_nfs --> off`, all we need to do is:

```
# setsebool ftpd_use_nfs on
# getsebool ftpd_use_nfs
ftpd_use_nfs --> on
```

These changes are however temporary in nature, and thus lost after a restart. To make these changes permanent, we need to use:

```
# setsebool -P ftpd_use_nfs on
# getsebool ftpd_use_nfs
ftpd_use_nfs --> on
```

This particular operation takes considerably more time since the policy has to be modified.

21.7 Analyzing SELinux Log Files

Understanding what is going wrong in a SELinux enabled environment isn't always easy, even though SELinux logs each occurrence of requests coming to it. To help us there are the **setroubleshoot** packages. Whether they're installed or not can be checked with:

```
# yum list installed | grep setrouble
setroubleshoot.x86_64                 3.2.28-3.el7             @anaconda
setroubleshoot-plugins.noarch         3.0.65-1.el7             @anaconda
setroubleshoot-server.x86_64          3.2.28-3.el7             @anaconda
```

All the events that have been logged by SELinux go to the *audit log*. In order for the audit log to be working, the *auditd* process needs to be started. We can confirm that it's working using `systemctl status auditd`, and if it is, we can view the log using:

```
# grep AVC /var/log/audit/audit.log
type=AVC msg=audit(1513680230.189:22): avc:  denied  { write } for  pid=709
 ↪  comm="accounts-daemon" name="root" dev="dm-0" ino=33574977
 ↪  scontext=system_u:system_r:accountsd_t:s0 tcontext=system_u:object_r:admin_home_t:s0
 ↪  tclass=dir
type=USER_AVC msg=audit(1513760499.935:10): pid=1 uid=0 auid=4294967295 ses=4294967295
 ↪  subj=system_u:system_r:init_t:s0 msg='avc:  received setenforce notice (enforcing=0)
 ↪  exe="/usr/lib/systemd/systemd" sauid=0 hostname=? addr=? terminal=?'
...
type=USER_AVC msg=audit(1513848001.387:320): pid=1 uid=0 auid=4294967295 ses=4294967295
 ↪  subj=system_u:system_r:init_t:s0 msg='avc:  received policyload notice (seqno=2)
 ↪  exe="/usr/lib/systemd/systemd" sauid=0 hostname=? addr=? terminal=?'
type=USER_AVC msg=audit(1513848601.536:329): pid=1 uid=0 auid=4294967295 ses=4294967295
 ↪  subj=system_u:system_r:init_t:s0 msg='avc:  received policyload notice (seqno=3)
 ↪  exe="/usr/lib/systemd/systemd" sauid=0 hostname=? addr=? terminal=?'
```

All SELinux messages start with the header **AVC**. Once such case where some action was denied by SELinux is:

```
# grep 'type=AVC' /var/log/audit/audit.log
type=AVC msg=audit(1513680230.189:22): avc:  denied  { write } for  pid=709
 ↪  comm="accounts-daemon" name="root" dev="dm-0" ino=33574977
 ↪  scontext=system_u:system_r:accountsd_t:s0 tcontext=system_u:object_r:admin_home_t:s0
 ↪  tclass=dir
```

The above incident tells us a file write system call was denied by SELinux on the directory `/root` as the context noted in the policy (*accountsd_t*) didn't match the context for the directory being accessed (*admin_home_t*). In the `/var/log/messages` file, more detail can be found on the event. If we check the `/var/log/messages` file, we can see the corresponding entry in it by searching for the term **sealert**:

```
# less /var/log/messages
Dec 19 16:13:56 vmPrime setroubleshoot: SELinux is preventing
↪  /usr/libexec/accounts-daemon from write access on the directory root. For complete
↪  SELinux messages run: sealert -l e277d205-f3b0-4ef7-a6c2-178a813da2e0
```

Finally, the noted command, `sealert -l e277d205-f3b0-4ef7-a6c2-178a813da2e0` explains the event in very great detail. **sealert** consults a database on the system to analyse what went wrong.

```
SELinux is preventing /usr/libexec/accounts-daemon from write access on the directory
↪  root.
*****  Plugin catchall (100. confidence) suggests   **************************
...
Additional Information:
Source Context                system_u:system_r:accountsd_t:s0
Target Context                system_u:object_r:admin_home_t:s0
Target Objects                root [ dir ]
Source                        accounts-daemon
Source Path                   /usr/libexec/accounts-daemon
Port                          <Unknown>
Host                          vmPrime.somuVMnet.com
Source RPM Packages           accountsservice-0.6.45-2.el7.x86_64
Target RPM Packages           filesystem-3.2-21.el7.x86_64
Policy RPM                    selinux-policy-3.13.1-166.el7.noarch
Selinux Enabled               True
Policy Type                   targeted
Enforcing Mode                Enforcing
Host Name                     vmPrime.somuVMnet.com
Platform                      Linux vmPrime.somuVMnet.com 3.10.0-693.el7.x86_64
.#1 SMP Tue Aug 22 21:09:27 UTC 2017 x86_64 x86_64
Alert Count                   1
First Seen                    2017-12-19 16:13:50 IST
Last Seen                     2017-12-19 16:13:50 IST
Local ID                      e277d205-f3b0-4ef7-a6c2-178a813da2e0
...
```

The confidence score suggests how likely a suggestion is to work. Note that these are automated attempts to solve whatever is wrong, and might not always be correct, and the SysAdmin must consider if it's a valid option and if the solution meets his/her requirements.

21.8 Configuring SELinux for Apache

This is how we deal with SELinux during real-life scenarios such as while configuring the apache web server. Let us consider we want to use a new document root at `/web`. We put an index.html file in the directory, and configure the `/etc/http/conf/httpd.conf` file with the new document root at `/web`, by adding the lines below:

```
DocumentRoot "/web"

```

```
3 <Directory "/web">
4 AllowOverride None
5 # Allow open access:
6 Require all granted
7 </Directory>
8
9 #<Directory "/var/www/html"> --> Manually commenting out the old tag start, and copying the original config with our own.
10 <Directory "/web">
11 ...
```

The lines 3-7 help provide access to the new document root. Note that this is modelled on the original `<Directory>` tag for the document root `/var/www`, which itself must not be edited or commented out (to stop other functionality from being disabled).

Once the *httpd.conf* file has been edited, we need to restart the apache service, with `systemctl restart httpd`. Now, the index.html should be available on the address `http://localhost`. Instead of the index.html page, we see an error page from the apache web server that reads *"The website you just visited is either experiencing problems or undergoing routine maintenance"*.

This generates the following SELinux notifications in the logs:

```
# In /var/log/audit/audit.log: [grep AVC /var/log/audit/audit.log]
type=AVC msg=audit(1513869456.604:322): avc:  denied  { getattr } for  pid=3683 comm="httpd" path="/web/index.html" dev="dm-0" ino=2556901 scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:default_t:s0 tclass=file

# In /var/log/messages: [less /var/log/messages ?sealert]
Dec 21 20:49:37 vmPrime setroubleshoot: SELinux is preventing httpd from getattr access on the file /web/index.html. For complete SELinux messages run: sealert -l 001da822-04b5-498d-8344-d78de9014597
```

It is clear from the message in the audit log that this is a case of context type mismatch. The source has a context label of *httpd_t* while the target directory and file (`/web/index.html`) have the context label of *default_t*. We can fix this using the command:

```
# semanage fcontext -a -t httpd_sys_content_t '/web(/.*)?'
# restorecon -R -v /web
restorecon reset /web context unconfined_u:object_r:default_t:s0->unconfined_u:object_r:httpd_sys_content_t:s0
restorecon reset /web/index.html context unconfined_u:object_r:default_t:s0->unconfined_u:object_r:httpd_sys_content_t:s0
```

Now when we visit the webpage at `http://localhost/index.html`, SELinux won't block us anymore.

Chapter 22

Configuring a Firewall

22.1 Understanding Firewall Configuration

The Linux kernel has a firewalling functionality called **netfilter**. In previous versions of RHEL, it used to be managed with **iptables**. However, now the default management interface is **firewalld** (even though iptables can still be used).

The design purpose of firewalld was to make firewall configuration easy, and this has been achieved with interfaces. Each of these interfaces is assigned a zone. There can be a private zone for private messages, where nothing is filtered, or a public zone for a server directly connected to the internet.

Next, services have to be connected to zones. Many services are already available by default and those that aren't are easy to configure and connect to the appropriate zone. Once these services are configured and are available, there are only a couple of command line utilities that we can use to setup our firewall.

22.2 Using Firewalld

To configure the Linux kernel firewall on RHEL 7, we use **firewalld**. While using iptables is still a valid option, it isn't the recommended way since many utilities write directly to firewalld. To ensure that everything is compatible, we should only use firewalld. To ensure that the firewalld service is running, we use:

```
# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset:
↪  enabled)
Active: active (running) since Fri 2017-12-22 10:29:51 IST; 18s ago
Docs: man:firewalld(1)
Main PID: 890 (firewalld)
CGroup: /system.slice/firewalld.service
└─890 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid

Dec 22 10:29:49 vmPrime.somuVMnet.com systemd[1]: Starting firewalld - dynami...
Dec 22 10:29:51 vmPrime.somuVMnet.com systemd[1]: Started firewalld - dynamic...
```

There are a couple of ways to add rules to the firewall. First there is the **firewall-cmd**, which is a command line utility to manage the firewall, and then there's **firewall-config**, a GUI utility which allows us to click to add services.

The basic configuration of a firewall in Linux is done with zones and services. To list all available zones and services we use:

```
# firewall-cmd --get-zones
block dmz drop external home internal public trusted work
# firewall-cmd --get-services
RH-Satellite-6 amanda-client amanda-k5-client bacula bacula-client bitcoin bitcoin-rpc bitcoin-testnet bitcoin-testnet-rpc ceph ceph-mon cfengine condor-collector ctdb dhcp dhcpv6 dhcpv6-client dns docker-registry dropbox-lansync elasticsearch freeipa-ldap freeipa-ldaps freeipa-replication freeipa-trust ftp ganglia-client ganglia-master high-availability http https imap imaps ipp ipp-client ipsec iscsi-target kadmin kerberos kibana klogin kpasswd kshell ldap ldaps libvirt libvirt-tls managesieve mdns mosh mountd ms-wbt mssql mysql nfs nrpe ntp openvpn ovirt-imageio ovirt-storageconsole ovirt-vmconsole pmcd pmproxy pmwebapi pmwebapis pop3 pop3s postgresql privoxy proxy-dhcp ptp pulseaudio puppetmaster quassel radius rpc-bind rsh rsyncd samba samba-client sane sip sips smtp smtp-submission smtps snmp snmptrap spideroak-lansync squid ssh synergy syslog syslog-tls telnet tftp tftp-client tinc tor-socks transmission-client vdsm vnc-server wbem-https xmpp-bosh xmpp-client xmpp-local xmpp-server
```

22.2.1 Default Zone

Now, if we need to find the default zone, the command is:

```
# firewall-cmd --get-default-zone
public
```

To set the default zone, the command is:

```
# firewall-cmd --set-default-zone home
success
# firewall-cmd --get-default-zone
home
```

22.2.2 Services

As far as the firewall is concerned, a service is a name assigned to a **protocol** and a **port**. And administrator can create his own services in `/etc/firewalld/services` directory. The default system services are stored in `/usr/lib/firewalld/services`. A typical service, such as the *high-availability.xml*, looks like:

```
<?xml version="1.0" encoding="utf-8"?>
<service>
<short>Red Hat High Availability</short>
<description>This allows you to use the Red Hat High Availability (previously named Red Hat Cluster Suite). Ports are opened for corosync, pcsd, pacemaker_remote, dlm and corosync-qnetd.</description>
<port protocol="tcp" port="2224"/>
<port protocol="tcp" port="3121"/>
<port protocol="tcp" port="5403"/>
```

```
<port protocol="udp" port="5404"/>
<port protocol="udp" port="5405"/>
<port protocol="tcp" port="21064"/>
</service>
```

This services binds multiple ports that we want open (for varied uses) to the TCP or UDP protocol. Another such complicated service is the *samba.xml* service, which is also a collection of ports:

```
<?xml version="1.0" encoding="utf-8"?>
<service>
<short>Samba</short>
<description>This option allows you to access and participate in Windows file and printer
↪   sharing networks. You need the samba package installed for this option to be
↪   useful.</description>
<port protocol="udp" port="137"/>
<port protocol="udp" port="138"/>
<port protocol="tcp" port="139"/>
<port protocol="tcp" port="445"/>
<module name="nf_conntrack_netbios_ns"/>
</service>
```

The last line, <module name="nf_conntrack_netbios_ns"/> states that for this service, a specific kernel module has to be loaded. Thus, if we want to create our own service in /etc/firewalld/services directory, it just needs to be contained within a valid XML file with the service tag, containing a short name, a description and port definition(s).

22.2.3 Adding services to zones

To add a service, we use the command:

```
# firewall-cmd --zone=home --add-service=high-availability
success
```

To get the configuration of the current zone, we use the command:

```
# firewall-cmd --list-all
home (active)
target: default
icmp-block-inversion: no
interfaces: ens33
sources:
services: ssh mdns samba-client dhcpv6-client high-availability
ports:
protocols:
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
```

To list all the services in a non-default zone, we use:

```
# firewall-cmd --zone=public --list-all
public
```

```
target: default
icmp-block-inversion: no
interfaces:
sources:
services: ssh dhcpv6-client
ports:
protocols:
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
```

Note that all services added in this manner are non-persistent and wiped with every reboot. To make them permanent, we just have to add the `--permanent` flag to each command:

```
# firewall-cmd --permanent --zone=home --add-service=high-availability
success
```

22.2.4 firewall-config

The firewall-config utility provides tabs of zones with a list of services in each, and the admin can check the services that should be available in each zone. The configuration can be set to either *runtime* or *permanent*.

Chapter 23

Configuring FTP Services

Chapter 24

Configuring Time Services

Chapter 25

Configuring VNC Access